▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 34

PASO A PASO



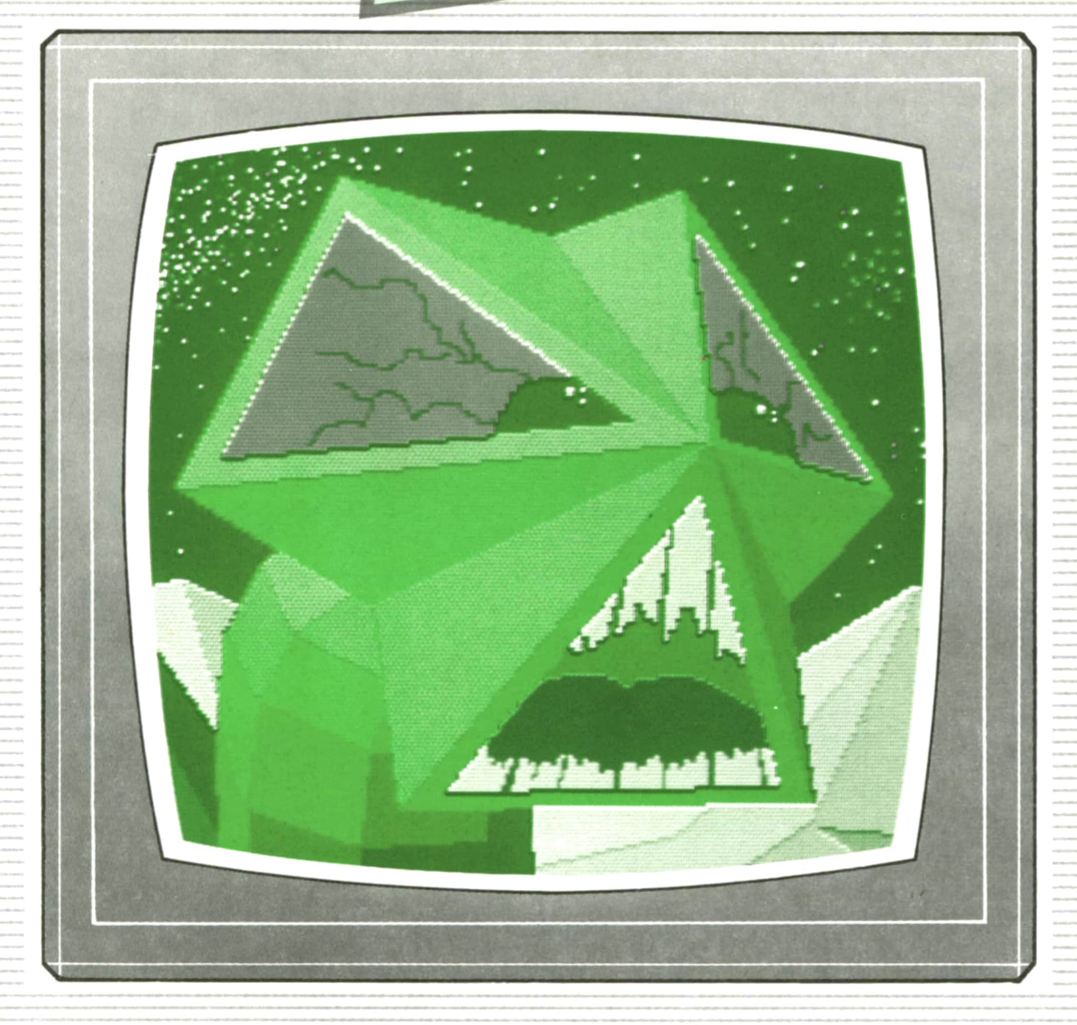
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 34

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación
y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

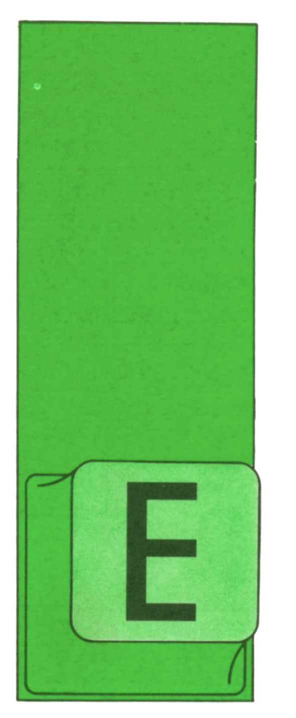
TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Jorge Thomas, Técnico en Informática. OTROS LENGUAJES (ADA): Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-185-1
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Marzo, 1988
P.V.P. Canarias: 335,–.

INDICE



INFORMATICA BASICA

FUNDAMENTOS DE LAS COMUNICACIONES



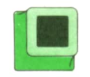
Introducción

EN los primeros ordenadores existían problemas muy graves en relación con la recogida y distribución de la información, dado que se usaban medios muy potentes de resolución de problemas que manejaban muchos datos, pero conseguir reunirlos resultaba costosísimo.

Para reunir o distribuir los datos e informaciones se emplearon en una primera etapa los medios convencionales de transporte o el correo, utilizándose últimamente los medios de comunicación, con transmisiones telegráficas, telefónicas o por otros medios más sofisticados.

El uso de sistemas de telecomunicaciones a la entrada y a la salida de los sistemas de computación ha mejorado notablemente la utilidad del ordenador. Mediante la red telefónica nacional o a través de líneas privadas de comunicaciones, las empresas, escuelas o individuos pueden tener acceso a ordenadores situados a grandes distancias. Esto ha sido también un factor influyente en la reducción de los costos del acceso a la información.

Como vimos en el capítulo titulado *Redes de ordenadores* cualquier tipo de información puede llegar a los lugares más remotos, haciendo posible su intercambio a nivel mundial. Así se ha aumentado la eficiencia en las transacciones comerciales, se han enriquecido los planes de estudio escolares y se ha incrementado notablemente la información disponible a cualquier ciudadano.



Características principales de un sistema de teleproceso

Los sistemas de computación de las décadas de los cuarenta y de los cincuenta tenían dos características fundamentales. En primer lugar, todas las unidades del sistema están ubicadas físicamente muy próximas unas de otras, den-

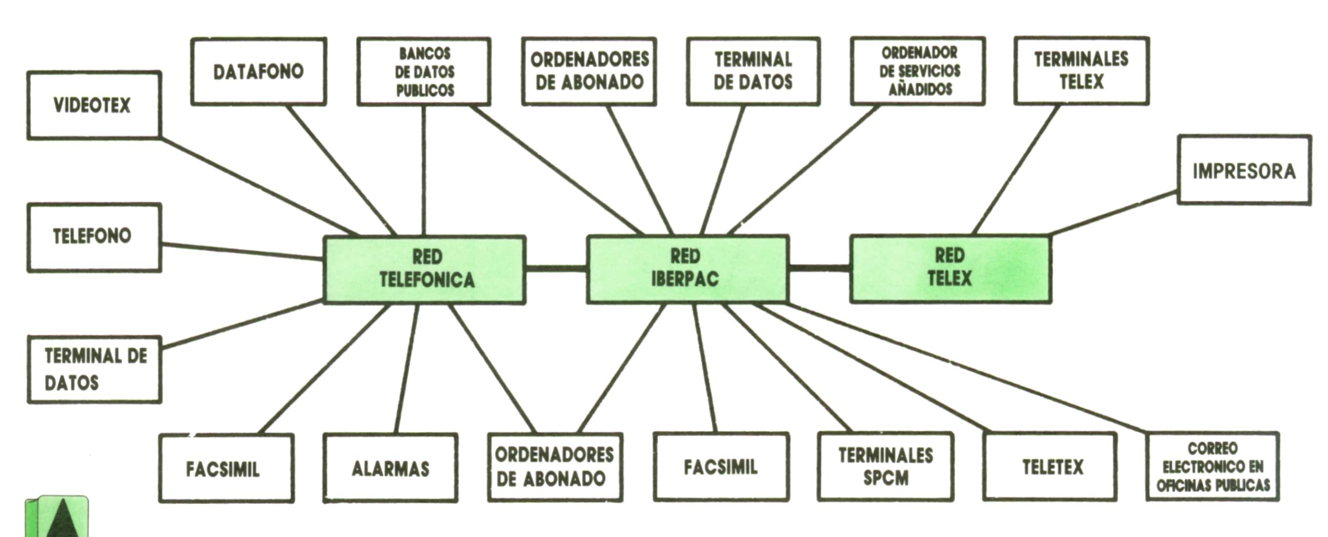
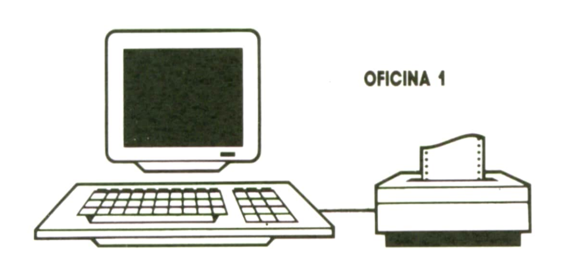


Fig. 1. Red telemática de servicios integrados (RSI).

tro de un radio, generalmente, no superior a 30 metros. La segunda peculiaridad consistía en que los grupos de tarjetas perforadas se agrupaban formando lotes o paquetes; esto tenía el inconveniente de que podían transcurrir horas o días hasta que se obtenía el listado impreso de los resultados. Procesando los programas por lotes, se intentaba obtener el máximo rendimiento del ordenador, pero no así la eficacia de las personas.



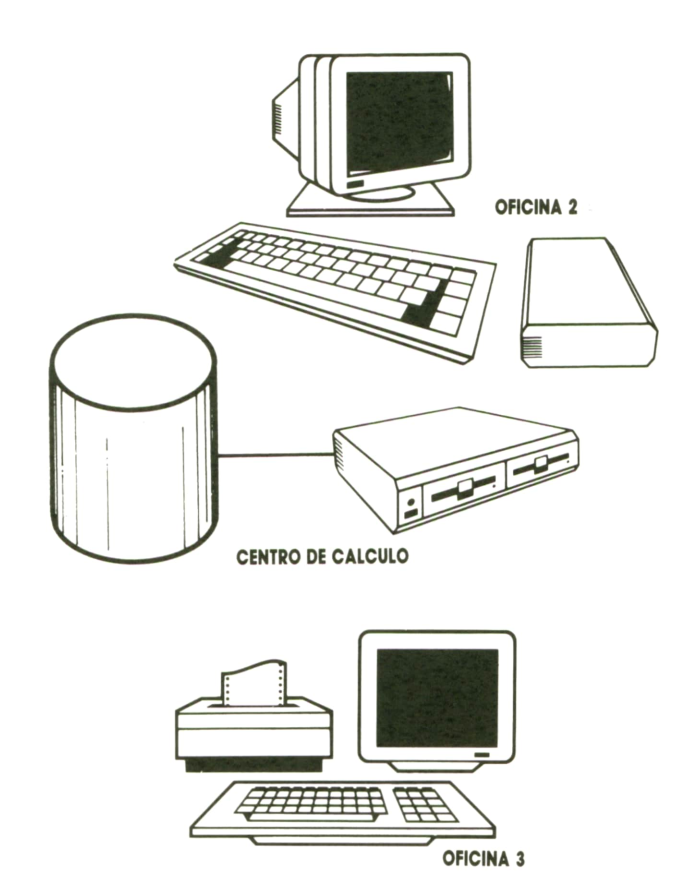




Fig. 2. En las décadas de los cincuenta y sesenta, todas las unidades estaban ubicadas físicamente próximas.

Sin embargo, aun procesando los programas por lotes, el ordenador estaba infrautilizado. En la década de los sesenta, de las técnicas de tiempo compartido, permitió la utilización por parte de varios usuarios de grandes sistemas de computación. Estas técnicas se basan en el aprovechamiento del tiempo. Mientras que utiliza el programa de un usuario se está procesando, se utiliza el ordenador para realizar la entrada/salida de los datos de otro.

En principio, las técnicas de proceso en tiempo compartido fueron pensadas para usuarios que estuviesen próximos unos a otros: en el campus de la Universidad, en un mismo edificio, etc. Pero al final de la década de los sesenta aparecieron unos dispositivos llamados *modems,* que permitían utilizar la línea telefónica para meter datos en el ordenador. Esta innovación hizo que los usuarios, tanto locales como residentes en lugares apartados, tuviesen acceso a la información. Con el modem puede utilizar el ordenador cualquier persona que disponga de un terminal y de una línea telefónica.

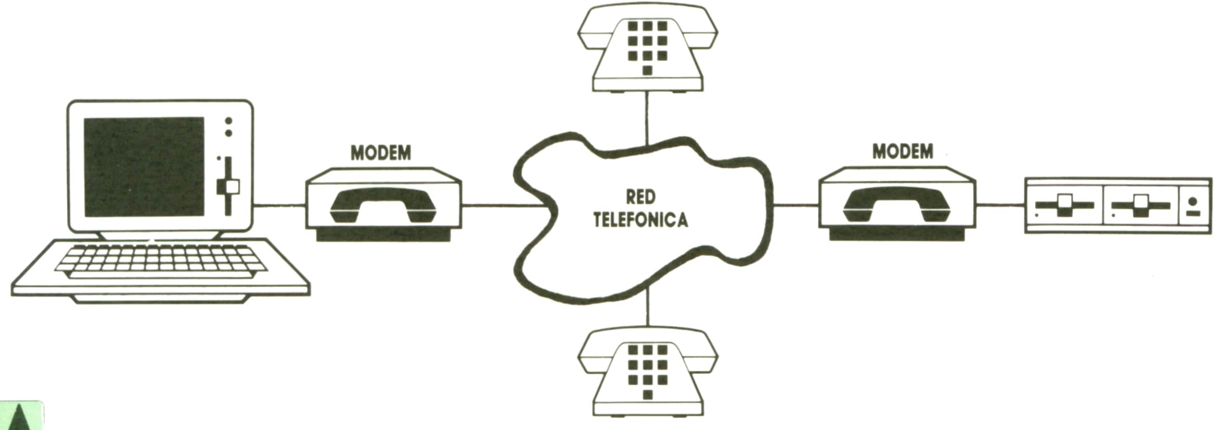




Fig.3 Con la aparición del MODEM, es posible transmitir señales digitales por un canal analógico.

Por tanto, las principales características de un sistema de teleproceso son las siguientes:

Entrada y salida remota. Los mensajes recibidos por el ordenador proceden de terminales remotos. Igualmente las respuestas desde dicho ordenador central se dirigen a los terminales remotos.

Entrada aleatoria. Cualquier terminal remoto puede acceder al ordenador central en cualquier momento.

Proceso inmediato. Es una de las grandes características que distinguen a los sistemas de teletratamiento de los sistemas de proceso convencional. En un sistema convencional, los datos se van recogiendo de algún medio de almacenamiento externo según llegan, y luego todas las correspondientes a un cierto período se procesan a la vez, como un lote. Por el contrario, en un sistema de teleproceso, cada transacción se procesa tan pronto como llega y de forma individual.

Tiempo de respuesta muy breve. Los sistemas que utilizan la facilidad de teleproceso deben diseñarse para que el tiempo de respuesta sea muy breve, con el fin de no congestionar el sistema central y la red de comunicaciones, ya que un sistema mal diseñado puede producir una interferencia notoria sobre el resto de la red.

ON-LINE. Este término describe un sistema de teleproceso en el que los datos que llegan desde los terminales remotos son introducidos directamente dentro del ordenador central, o en el que los datos ya procesados son transmitidos directamente desde el ordenador a la localidad donde serán utilizados.

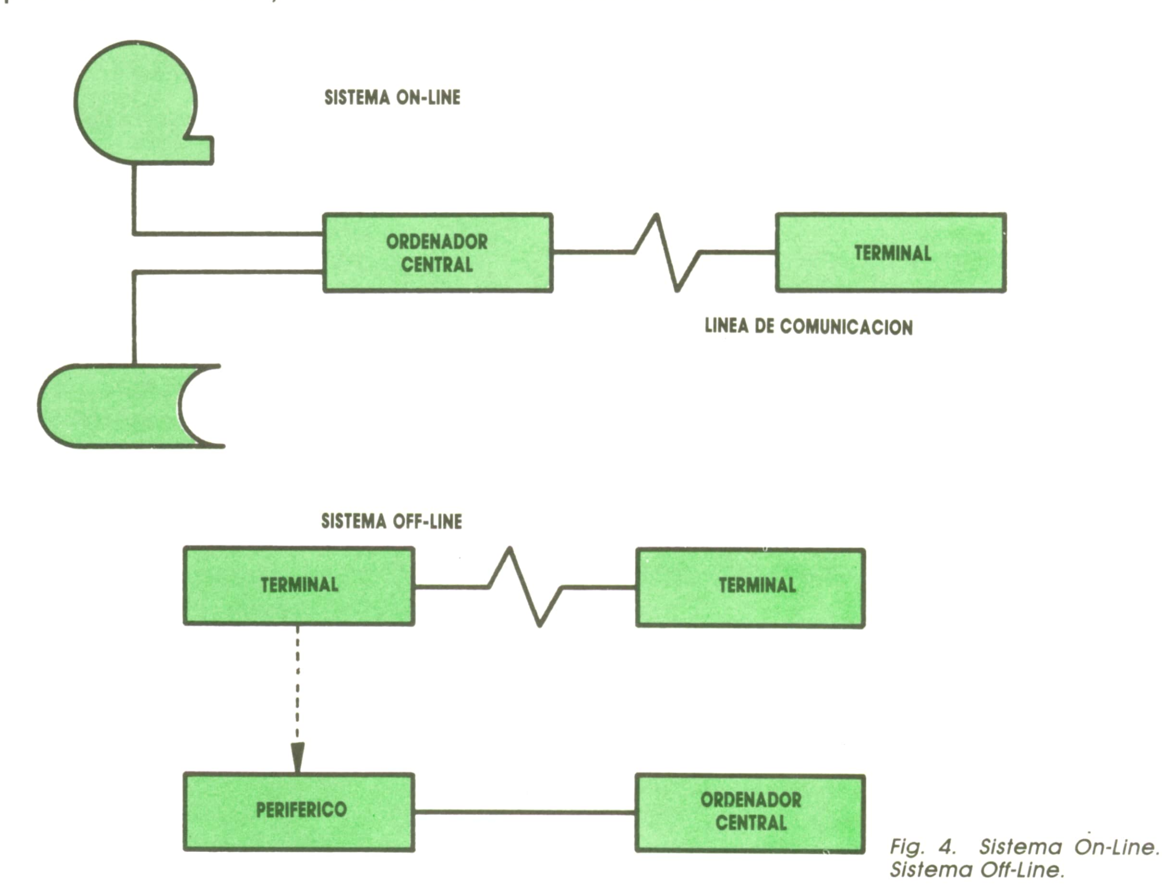


Fig. 4. Sistema On-Line. Sistema Off-Line.

La forma de comunicación establece la diferencia entre los sistemas *On-Line* y *Off-Line*. En el primero los datos llegan directamente al ordenador y se tratan directamente constituyendo, por tanto, un sistema de tratamiento a distancia o "teletratamiento" que también permitirá devolver resultados. En los sistemas *Off-Line* los datos los recibe un terminal, por ejemplo, una lectora de cinta. Después se coloca esa ficha en el lector para proporcionar los datos al ordenador. Evidentemente, un sistema *On-Line* precisa un hardware y software más elaborados, la rapidez del proceso es mayor y el ordenador puede efectuar mayor número de controles.

Conmutación

Se conoce por conmutación al conjunto de funciones que se llevan a cabo para permitir establecer una comunicación entre terminales de una red. Entre las funciones a realizar para permitir el encaminamiento de la información de un terminal a otro se encuentran:

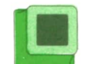
— El direccionamiento: para codificar la información de forma que se identifique el destinatario.
— El establecimiento de la ruta de datos a través de la red mediante la función de conmutación apropiada.
— La gestión de prioridades: basada en conceptos como volúmenes de tráfico de datos, haciendo que ciertos terminales tengan más prioridad que otros en la comunicación.

Términos como conmutación de paquetes son muy frecuentes cuando se habla de transmisión. Veamos un poco de qué se trata.

Hablábamos anteriormente del establecimiento de la ruta de datos, bueno, pues se puede realizar mediante:

— Conmutación de mensajes: en esta modalidad de conmutación, los mensajes son enviados completos desde el terminal origen a la red; ésta los almacena y posteriormente los envía al terminal destino.

— Conmutación de paquetes: es similar a la conmutación de mensajes, con la diferencia de que la información transmitida entre terminales se fracciona en unidades menores llamadas paquetes, cuyo tamaño está previamente determinada. Los paquetes son enviados a la red independientemente.



Otras definiciones

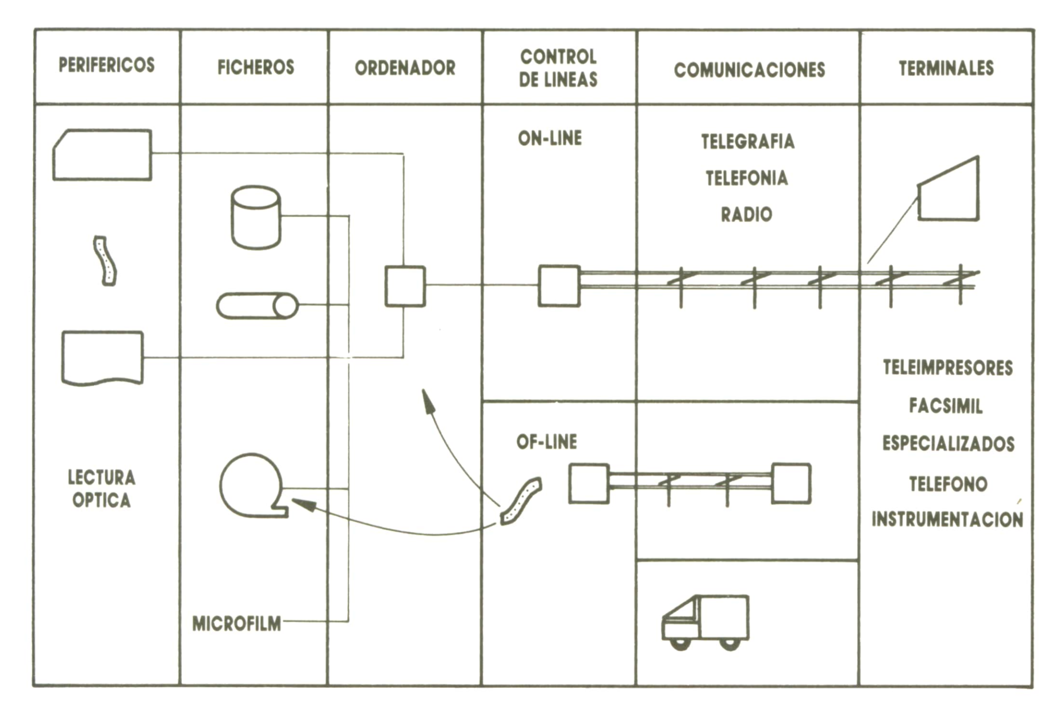


Fig. 5 Hardware de teleinformática.

En un sistema de teletratamiento se llama *tiempo de respuesta* al que transcurre entre la transmisión del último carácter de la pregunta y la recepción del primer carácter de la respuesta.

Un sistema en *tiempo real* es aquel cuyo tiempo de respuesta es suficientemente pequeño como para que la respuesta influya en el proceso que desencadenó la pregunta. En sistemas en los cuales el tiempo de respuesta es del orden de algunos minutos, se suele utilizar el término de *tiempo útil* en vez de *tiempo real.*

Según el tipo de aplicación, existen diferentes clases de Sistemas de Teletratamiento, entre los que podemos citar:

— *De tiempo compartido.* Varios usua-

rios comparten el ordenador en tiempo real.

— *De consulta.* Un banco de datos al que tienen acceso múltiples usuarios.

— *De recogida de dátos.* Pueden ser o no de tiempo real, y normalmente tienen fines estadísticos.

— *De distribución de información.*

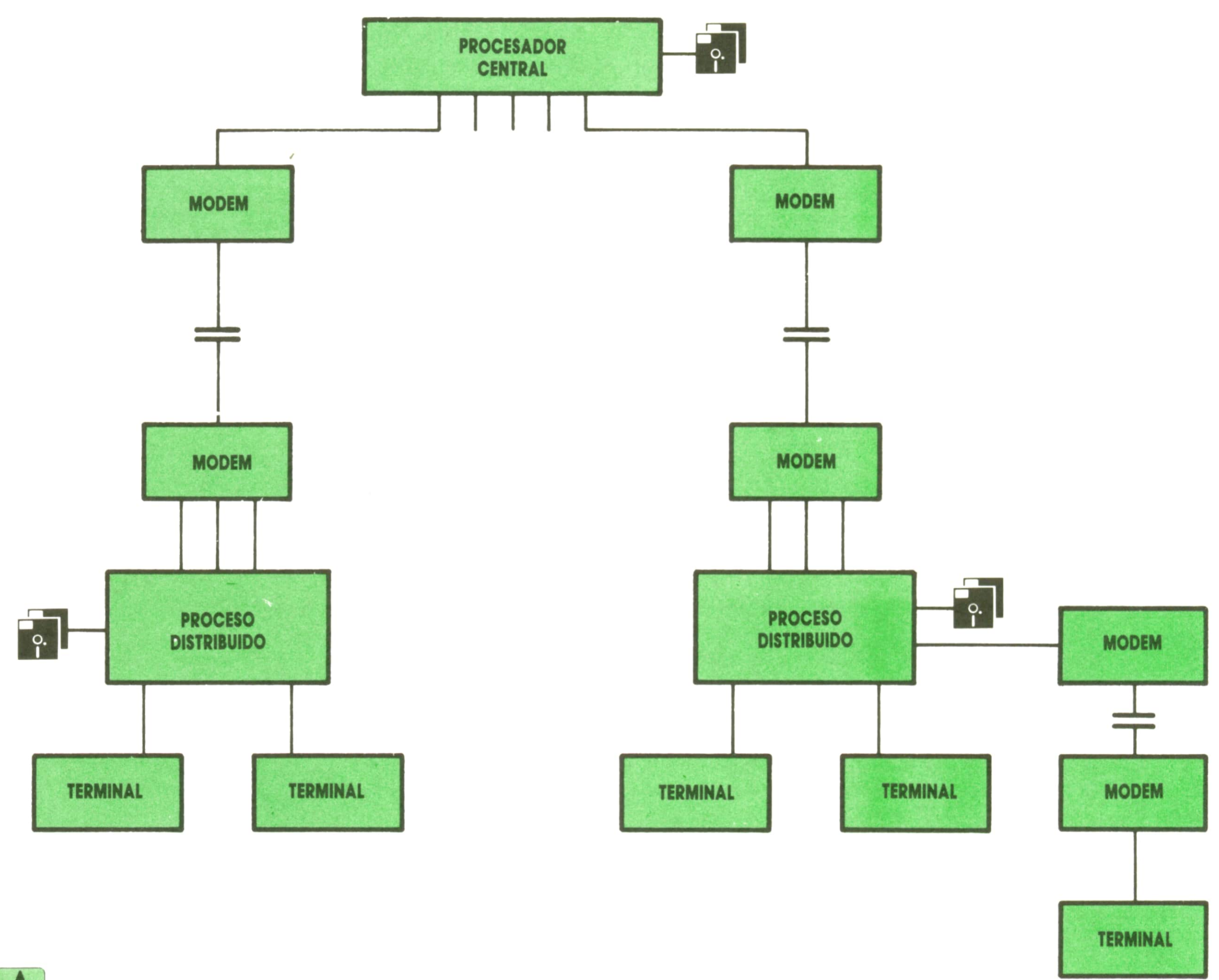
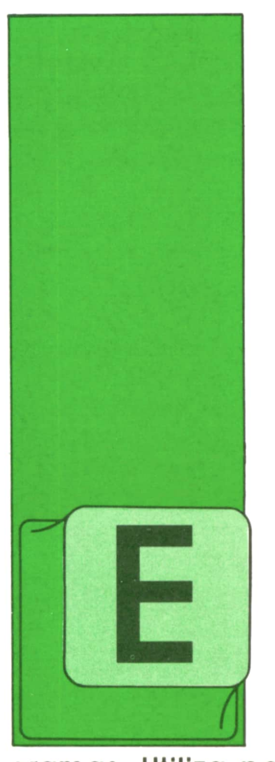


Fig. 6. Distribución de la información.

MAQUINA 8088



Interrupciones del DOS



EL sistema operativo DOS ("Disk Operating System") tiene asociadas a un conjunto de interrupciones software las tareas más comunes que se realizan en los programas. Utiliza para ello las interrupciones 32 a 39 (20 a 27 hexadecimal) y tiene reservadas para uso futuro de la 40 a la 63 (28 a 3F hexadecimal).

El software que atiende a estas interrupciones es una parte importante del núcleo del DOS y, como ya se ha dicho, se carga en memoria en la fase de IPL.

A continuación vamos a describir brevemente las tareas realizadas por cada interrupción:

INT 20H. Terminación de programas. Cuando termina un programa, el control de ejecución debe volver al DOS, pero antes deben realizarse las acciones necesarias para restaurar el estado en que se encontraba el DOS cuando el programa comenzó. Estas acciones se llevan a cabo ejecutando la instrucción INT 20H. Se requiere que el registro CS tenga el mismo valor que cuando comenzó el programa.

INT 2 1 H. "Function Calls". Se denomina así a una larga serie de funciones que proporciona el DOS referentes a:

— Lectura y escritura de caracteres.
— Gestión de ficheros.
— Gestión de memoria.
— Definición y petición de fecha y hora.
— Ejecución de otros programas.
— Etc.

Las llamadas a estas funciones se realizan definiendo el código de la función deseada en el registro AH y ejecutando a continuación la instrucción INT 21H. Los datos que hay que suministrar a las funciones y los que éstas devuelven se transfieren por medio de los registros del 8088.

Debido al gran número de funciones asociadas a esta interrupción, se le dedica más adelante un apartado completo a describir las funciones más interesantes.

Dirección de terminación. La interrupción 22H no es una interrupción como las demás y no debe ejecutarse nunca la instrucción INT 22H. El DOS utiliza su apuntador en el vector de interrupciones (en la dirección de memoria 0000:0088 hexadecimal), para guardar ahí la dirección de terminación de los programas.

Si desde un programa A se quiere desencadenar la ejecución de otro programa B, antes de transferir el control a B hay que definir en 0000:0088 la dirección a donde debe volver el DOS cuando dé por terminado el programa B.

Dirección de "Ctrl-Break". La interrupción 23H tampoco es una interrupción como las demás y no debe ejecutarse nunca la instrucción INT 23H.

Explicaremos brevemente la problemática relacionada con esta interrupción. El DOS tiene previsto que se puedan ejecutar acciones especiales cuando en el teclado se pulsen simultáneamente las teclas "Ctrl" y "Break". Por ello, cuando el DOS detecta la citada combinación de teclas, ejecuta internamente la instrucción INT 23H.

Por omisión, una rutina del propio DOS atiende a dicha interrupción, pero si un programa requiere alguna acción específica de "Ctrl-Break" puede colocar el apuntador al código que realiza dicha acción en la dirección de memoria 0000:008C hexadecimal. (Que es la di-

rección correspondiente a esta interrupción.)

Por otra parte, si desde un programa A se quiere desencadenar la ejecución de otro programa B, que deba realizar acciones específicas de "Ctrl-Break", antes de transferir el control a B hay que definir en la dirección 0000:008C el apuntador al código que realice dicha acción.

INT 24H. Tratamiento de errores críticos. Cuando ocurre un error de los considerados "críticos", como:

— Intento de escritura sobre diskette protegido.
— Unidad de diskette no preparada.
— Falta de papel en la impresora.
— Etc.

el DOS ejecuta internamente una instrucción INT 24H, dando la oportunidad para que en los programas que se desee se pueda hacer un tratamiento especial de dichos errores. Para ello debe asociarse la rutina que realiza dicho tratamiento a esta interrupción. Es decir, debe escribirse un apuntador a dicha rutina en la dirección de memoria 0000:0090 (hex).

Por omisión, la acción la efectúa una rutina del propio DOS.

INT 25H. Lectura absoluta de disco. Esta interrupción del DOS utiliza el BIOS para realizar la lectura de los sectores especificados de un disco o diskette, ignorando totalmente la organización lógica del mismo (directorios, ficheros, etc.). Para efectuar la lectura, debe definirse:

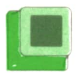
— El número de la unidad en el registro AL, es decir, 0 para la unidad A, 1 para la B, etc.
— El número lógico del primer sector en el registro DX.
— El número de sectores en el registro CX.
— La dirección del área de lectura en los registros DS:BX.

INT 26H. Escritura absoluta en disco. Esta interrupción utiliza igualmente el BIOS, para realizar la escritura de uno o más sectores de un disco o diskette. Para ello deben definirse los registros de la misma forma que en la interrupción anterior.

INT 27H. Terminación de un programa residente. Cuando se desea dejar un programa residente en memoria, debe terminarse dicho programa ejecutando la instrucción INT 27H, habiendo definido en los registros CS:DX la dirección del byte siguiente al último que debe ser respetado.

A partir de ese momento, el DOS considera a este programa como una extensión suya, respetando la memoria que ocupa.

Puede haber varias razones para que un programa deba mantenerse permanentemente en memoria, pero el caso más frecuente es que dicho programa esté diseñado para atender alguna interrupción software.



Interrupción 21H

Por medio de esta interrupción se puede ejecutar en gran número de funciones suministradas por el DOS.

Antes de pasar a describir las funciones más interesantes conviene reseñar los siguientes puntos:

— "Standard input" es el dispositivo que se ha definido al DOS para que lo utilice como origen de los datos de determinadas funciones. Por omisión el "standard input" es el teclado, pero el DOS permite cambiarlo por cualquier dispositivo del que se pueda leer, como, por ejemplo: línea de comunicaciones, fichero, etc. (ver "Redirección de los dispositivos standard" en el manual del DOS).

— "Standard output" es el dispositivo que se ha definido al DOS para que lo utilice como destino de los datos de determinadas funciones. Por omisión el "standard output" es la pantalla, pero el DOS permite definir en su lugar cualquier dispositivo sobre el que se pueda escribir, como, por ejemplo: impresora, línea de comunicaciones, fichero, etc.

— "Standard printer" es el dispositivo que se ha definido al DOS para que lo utilice como destino de los datos de determinadas funciones. Por omisión el "Standard printer" es la impresora, pero con el comando MODE del DOS puede cambiarse para que sea una línea de comunicaciones.

— La interrupción 21H dispone de dos métodos de gestión de ficheros. El método que podemos denominar «tradicional», basado en el concepto de FCB ("File Control Block") y el método "nuevo" (que es el único que vamos a descri-

bir) en el que sólo es necesario especificar el "ASCIIZ" del fichero deseado.

— Se denomina "ASCIIZ" de un fichero a una cadena de caracteres ASCII donde se especifica secuencialmente:

- — La unidad de disco (opcional).
- — El "path" del directorio.
- — El nombre del fichero.
- — La extensión.
- — El carácter ASCII nulo (00H).

— El flag de acarreo (CF o "carry flag") desempeña un papel importante en las funciones relacionadas con los ficheros, ya que sirve para indicar si la función se ha realizado correctamente o si, por el contrario, se ha detectado algún error.

Si se detecta algún error, la función devuelve el flag de acarreo en "on" (valor 1) y el código de error en el registro AX. En caso contrario, se devuelve dicho flag en "off" (valor 0).

— Se denomina "Handle" de un fichero a un código devuelto por las funciones de creación (3CH) o de apertura (3DH) que se usará para referenciar de forma concisa al fichero creado o abierto en las siguientes operaciones (lectura, escritura, etc.) que se realicen con él, hasta que dicho fichero sea cerrado (función 3EH).

— El DOS maneja para cada fichero abierto un indicador de la posición sobre la que debe aplicar la siguiente operación de lectura o escritura que se denomina "apuntador de lectura/escritura". El valor del apuntador se actualiza en cada operación y puede ser definido explícitamente con la función 42H.

— Para cada fichero que abre, el DOS reserva una zona de memoria que le sirve de área auxiliar de trabajo para las operaciones de lectura y escritura, que denominaremos "buffers del fichero".

A continuación vamos a describir algunas de las funciones que pueden realizarse con la interrupción 21H, indicando para cada una el código que debe definirse en el registro AH:

AH=1. Lectura de un carácter. Esta función espera hasta que se defina un carácter en el dispositivo definido como "standard input"; una vez recibido, lo reproduce sobre el "standard output" y termina la función devolviendo el carácter en el registro AL.

AH=2. Representación de un carácter. El carácter recibido en el registro DL se reproduce sobre el "standard output".

AH=5. Impresión de un carácter. El carácter recibido en el registro DL se imprime sobre el "standard printer".

AH=9. Representación de una cadena de caracteres. Al llamar a esta función, la pareja de registros DS:DX deben contener la dirección de un área de memoria que contenga la cadena de caracteres que se quiere representar sobre el dispositivo definido como "standard output". La cadena de caracteres debe terminarse con el carácter $. En el programa puesto como ejemplo en el tomo 10 se utiliza esta función.

AH=0AH. Lectura de una cadena de caracteres. En el momento de la llamada, la pareja de registros DS:DX deben contener la dirección del área de memoria sobre la que se quiere recibir la cadena de caracteres. El primer byte de dicha área debe contener su longitud (que debe no ser cero). Esta función lee caracteres del "standard input" y los coloca en el área especificada a partir del tercer byte, ya que en el segundo devuelve el número de caracteres recibidos.

AH=3CH. Creación de un fichero. La pareja de registros DS:DX debe contener la dirección de un área de memoria que contenga el ASCIIZ del fichero que se desea crear. El atributo es un código que indica el uso que se le va a dar al fichero y debe especificarse en el registro CX. Debe definirse CX=0 para los ficheros de lectura/escritura. CX=1 para los de sólo lectura, CX=2 para los ficheros ocultos y CX=4 para los ficheros del sistema.

Si el fichero especificado no existe, se crea y se abre. Si existe y es de escritura, se abre para ser reescrito desde el origen. Si existe y es de sólo lectura, se produce el error número 5.

Cuando la función se ejecuta normalmente (flag de acarreo en off), se devuelve el "handle" del fichero en el registro AX.

AH=3DH. Apertura de un fichero. La pareja de registros DS:DX debe contener la dirección de un área de memoria que contenga el ASCIIZ del fichero que se desea abrir.

El código de acceso debe definirse en el registro AL. Dicho código indica si el fichero va a ser abierto sólo para lectura

(AL=0), sólo para escritura (AL=1) o para lectura y escritura (AL=2).

Cuando la función se ejecuta normalmente (flag de acarreo en off), se devuelve el "handle" del fichero en el registro AX, y el "apuntador de lectura/escritura" queda apuntando al primer byte del fichero.

AH=3EH. Cierre de un fichero. Esta función cierra el fichero cuyo "handle" se haya especificado en el registro BX y libera los "buffers" que se le reservaron en la apertura.

AH=3FH. Lectura de un fichero. El registro BX debe contener el "handle" del fichero del que se desea leer, CX debe contener el número de bytes deseados, y los registros DS:DX deben contener la dirección del área de memoria sobre la que se desea leer dichos bytes.

La lectura se realiza a partir de la posición definida en el "apuntador de lectura/escritura", devolviendo la función en el registro AX el número de bytes realmente leídos, que puede no coincidir con los solicitados si en la operación se llega al fin del fichero.

AH=40H. Escritura sobre un fichero. En el registro BX debe especificarse el "handle" del fichero, en DS:DX debe estar la dirección del área de memoria que contiene los bytes a escribir y en CX el número de ellos.

La escritura se realiza a partir de la posición definida en el "apuntador de lectura/escritura" devolviendo la función en el registro AX el número de bytes realmente escritos, que puede no coincidir con los solicitados si durante la operación se ha llenado el disco.

AH=41H. Borrado de un fichero. La pareja de registros DS:DX debe contener la dirección de un área de memoria que contenga el ASCIIZ del fichero que se desea borrar.

Los ficheros que tengan el atributo de "sólo lectura" no pueden borrarse. Para borrar un fichero de este tipo, hay que utilizar la función 43H para cambiarle previamente el atributo.

AH=42H. Posicionamiento en un fichero. Esta función sirve para modificar el "apuntador de lectura/escritura". Para ello, el "handle" del fichero debe especificarse en el registro BX y el "desplazamiento" del "apuntador de lectura/escritura" en los registros CX:DX (la parte más significativa debe ir en el registro CX). Dicho "desplazamiento" puede interpretarse de tres formas diferentes, dependiendo del contenido de AL:

— Si AL=0, se interpreta que el "desplazamiento" es absoluto, es decir, se cuenta desde el origen del fichero.

— Si AL=1, se interpreta que el "desplazamiento" es relativo, es decir, se cuenta desde la posición previa del "apuntador de lectura/escritura".

— Si AL=2, se interpreta que el "desplazamiento" está referido al final del fichero.

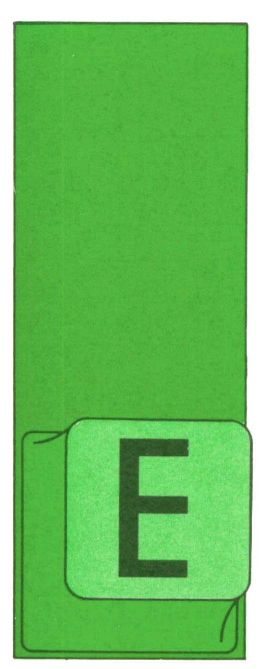
Si la función termina correctamente, se obtiene en DX:AX el nuevo valor del "apuntador de lectura/escritura".

AH=43H. Lectura/Cambio del atributo de un fichero. Esta función sirve para las dos cosas, para leer y para cambiar el atributo de un fichero. Para leerlo hay que definir AL=0, en cuyo caso al terminar la función se obtiene el atributo en el registro CX. Para escribirlo hay que definir AL=1, y el atributo deseado en el registro CX.

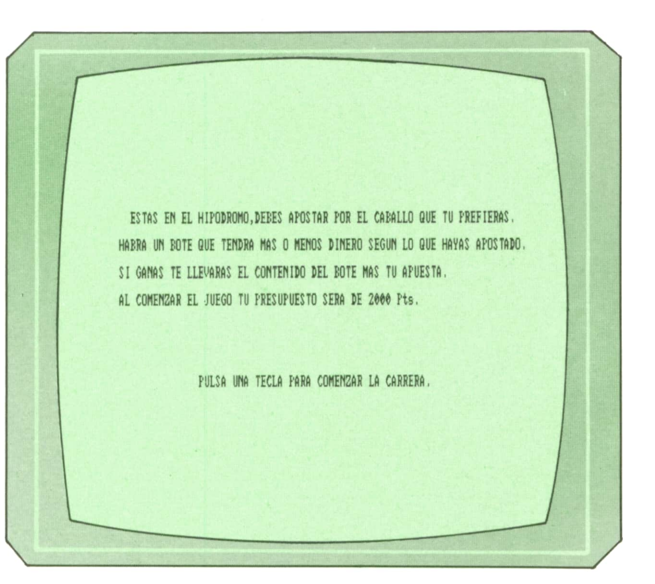
En cualquier caso, la pareja de registros DS:DX debe contener la dirección del área de memoria que contenga el ASCIIZ del fichero en cuestión.

PROGRAMAS

Programa: Carrera de caballos



STE programa, que ya apareció en su versión para SPECTRUM, está realizado bajo GWBASIC en un IBM; para que pueda funcionar en el AMSTRAD hay que realizar los siguientes cambios:



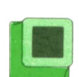


Fig. 1. Presentación.

```
PROGRAMA: CARRERA DE CABALLOS
=============================

100 REM ***************************************************
101 REM *                                                 *
102 REM *  ***   ***  ****   ***  *     *      ***   ***  *
103 REM * *   * *   * *   * *   * *     *     *   * *   * *
104 REM * *     *   * *   * *   * *     *     *   * *     *
105 REM * *     *   * ****  *   * *     *     *   *  ***  *
106 REM * *     ***** *   * ***** *     *     *   *     * *
107 REM * *   * *   * *   * *   * *     *     *   * *   * *
108 REM *  ***  *   * ****  *   * ***** *****  ***   ***  *
109 REM *                                                 *
110 REM ***************************************************
111 REM
112 REM ***********************
113 REM * AUTOR: CARLOS DORAL *
114 REM ***********************
115 REM
116 REM ********************************
117 REM **** (c) Ed. Siglo Cultural ****
118 REM **** (c) 1987               ****
119 REM ********************************
120 REM
121 FOR F=1 TO 80
122     LET A$=A$+CHR$(176)
123 NEXT F
124 LET E$=CHR$(174)
125 DIM C(10)
126 DIM F$(10)
127 FOR F=1 TO 10
128     LET F$(F)=E$+STR$(F)
129 NEXT F
130 CLS
131 LOCATE 8,6
132 PRINT "ESTAS EN EL HIPODROMO,DEBES APOSTAR POR EL CABALLO QUE TU PREFIERAS."
```

```
133 PRINT
134 PRINT "   HABRA UN BOTE QUE TENDRA MAS O MENOS DINERO SEGUN LO QUE HAYAS APO
STADO."
135 PRINT
136 PRINT "   SI GANAS TE LLEVARAS EL CONTENIDO DEL BOTE MAS TU APUESTA."
137 PRINT
138 PRINT "   AL COMENZAR EL JUEGO TU PRESUPUESTO SERA DE 2000 Pts."
139 LOCATE 20,18
140 PRINT "PULSA UNA TECLA PARA COMENZAR LA CARRERA."
141 LOCATE 20,18
142 PRINT SPACE$(62)
143 LET K$=INKEY$
144 IF K$="" THEN GOTO 139
145 CLS
146 LET PT=2000
147 LOCATE 10,30
148 PRINT "TIENES ";PT;" Pts."
149 LOCATE 20,10
150 PRINT "( CUANTO VAS A APOSTAR (MAX.";PT;"Pts.) ";
151 INPUT "";B$
152 IF LEN (B$)<1 OR LEN(B$)>LEN(STR$(PT)) OR ASC(B$)<48 OR ASC(B$)>57 THEN LOCA
TE 20,51:PRINT SPACE$(LEN (B$)+1):GOTO 149
153 LET AP=VAL(B$)
154 IF AP<1 OR AP>PT THEN LOCATE 20,51:PRINT SPACE$(LEN(B$)+1):GOTO 149
155 FOR F=1 TO 10
156    LET C(F)=76
157 NEXT F
158 CLS
159 RANDOMIZE TIMER
160 LET BO=INT(RND*AP)
161 LOCATE 10,26
162 PRINT "EL BOTE ES DE ";BO;" Pts."
163 FOR F=1 TO 3000
164 NEXT F
165 CLS
166 PRINT "ELIGE CABALLO (1-10) ";
167 INPUT "";B$
168 IF LEN (B$)<1 OR LEN(B$)>2 OR ASC(B$)<48 OR ASC(B$)>57 THEN GOTO 165
169 LET NC=VAL (B$)
170 IF NC<1 OR NC>10 THEN GOTO 165
171 CLS
172 GOSUB 190
173 LET I=1
174 FOR F=2 TO 20 STEP 2
175    LOCATE F,76
176    PRINT F$(I)
177    LET I=I+1
178 NEXT F
179 LOCATE 18,25
180 PRINT "PULSA RETURN PARA DAR LA SALIDA"
181 LET K$=INKEY$
182 IF K$<>CHR$(13) THEN GOTO 181
183 LOCATE 18,25
184 PRINT SPACE$(31)
185 RANDOMIZE TIMER
186 LET RN=1+INT(RND*10)
187 GOSUB 204
188 GOTO 185
189 REM
190 REM ***********************
191 REM * DIBUJO DEL DECORADO *
192 REM ***********************
193 REM
194 FOR F=1 TO 80 STEP 2
195    LOCATE 1,F
196    PRINT CHR$(176)
197 NEXT F
```

```
198 FOR F=3 TO 21 STEP 2
199    LOCATE F,1
200    PRINT A$
201 NEXT F
202 RETURN
203 REM
204 REM *****************************
205 REM * IMPRESION DE LOS CABALLOS *
206 REM *****************************
207 REM
208 LET X=C(RN)
209 LET Y=RN*2
210 LET X=X-1
211 LOCATE Y,X
212 PRINT F$(RN)
213 LOCATE Y,X+LEN(F$(RN))
214 PRINT " "
215 LET C(RN)=X
216 IF X=1 AND NC=RN THEN GOTO 220
217 IF X=1 AND NC<>RN THEN GOTO 244
218 RETURN
219 REM
220 REM *********************
221 REM * GANA TU CABALLO ! *
222 REM *********************
223 REM
224 LOCATE 21,20
225 PRINT " - - ENHORABUENA , HA GANADO TU CABALLO ! ! "
226 FOR C=1 TO 6
227    FOR F=900 TO 100 STEP -20
228       SOUND F,.005
229    NEXT F
230    LOCATE 21,20
231    PRINT SPACE$(44)
232    LOCATE 21,20
233    PRINT " - - ENHORABUENA , HA GANADO TU CABALLO ! ! "
234 NEXT C
235 SOUND F,.1
236 CLS
237 LOCATE 10,14
238 PRINT "TENIAS ";PT;" Y AHORA GANAS ";AP;" + ";BO;" = ";PT+AP+BO;" Pts."
239 LET PT=PT+AP+BO
240 LOCATE 12,30
241 LET A=1
242 GOTO 260
243 REM
244 REM **********************
245 REM * PIERDES LA CARRERA *
246 REM **********************
247 REM
248 LOCATE 21,22
249 PRINT " OOOOH, - QUE LASTIMA !,HAS PERDIDO "
250 PLAY "L4 O3 GFED O2 L2 C"
251 LET PT=PT-AP
252 CLS
253 IF PT>0 THEN LET A=1:LOCATE 10,30:PRINT "TE QUEDAN ";PT;" Pts.":LOCATE 12,25
:GOTO 260
254 IF PT<1 THEN LOCATE 10,15:PRINT "SIENTO DECIRTE QUE NO TE QUEDA MAS DINERO."
255 FOR F=1 TO 2000
256 NEXT F
257 LET A=2
258 LOCATE 12,25
259 REM
260 REM *********************************
261 REM * QUIERES JUGAR OTRA VEZ? (S/N) *
262 REM *********************************
263 REM
```

```
264 PRINT " ( QUIERES SEGUIR JUGANDO (S/N) ? "
265 LET K$=INKEY$
266 IF A=1 THEN IF K$="S" OR K$="s" THEN CLS:GOTO 147
267 IF A=2 THEN IF K$="S" OR K$="s" THEN GOTO 130
268 IF K$="N" OR K$="n" THEN END
269 GOTO 265
```

AMSTRAD:

```
122 LET A$=A$+CHR$(207)
124 LET E$=CHR$(242)
130 MODE 2
131 LOCATE 6,8
139 LOCATE 18,20
141 LOCATE 18,20
147 LOCATE 30,10
149 LOCATE 10,20
152 IF LEN/B$<1 OR LEN(B$)>LEN(STR(PT)) OR ASC(B$)<48 OR ASC(B$)>57 THEN LOCATE 51,20:PRINT SPACE$(LEN(B$)+1):GOTO 149
154 IF AP<1 OR AP>PT THEN LOCATE 51,20:PRINT SPACE$(LEN(B$)+1):GOTO 149
159 RANDOMIZE TIME
160 LET BO=INT(RND(1)*AP)
161 LOCATE 26,10
175 LOCATE 76,F
179 LOCATE, 25,18
183 LOCATE 25,18
185 RANDOMIZE TIME
186 LET RN=1+INT(RND(1)*10)
195 LOCATE F,1
199 LOCATE 1,F
211 LOCATE X,Y
224 LOCATE 20,21
228 REM
230 LOCATE 20,21
323 LOCATE 20,21
235 REM
237 LOCATE 14,10
240 LOCATE 30,12
248 LOCATE 22,21
250 REM
253 IF PT>0 THEN LET A=1:LOCATE 30,10:PRINT "TE QUEDAN";PT;" PTS.":LOCATE 25,12:GOTO 260
254 IF PT<1 THEN LOCATE 15,10:PRINT "SIENTO DECIRTE QUE NO TE QUEDA MAS DINERO."
258 LOCATE 25,12
```

Este programa simula una carrera de caballos donde tú podrás apostar el dinero que quieras por alguno de los caballos.

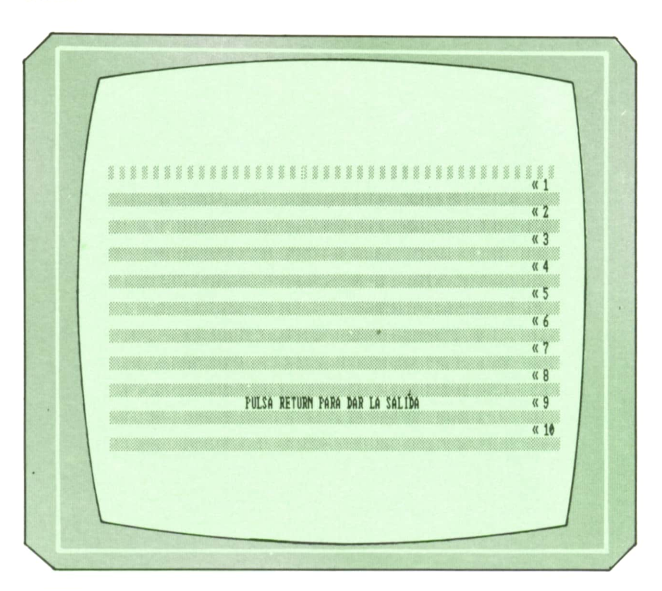


Fig. 3. Llegada a la meta.

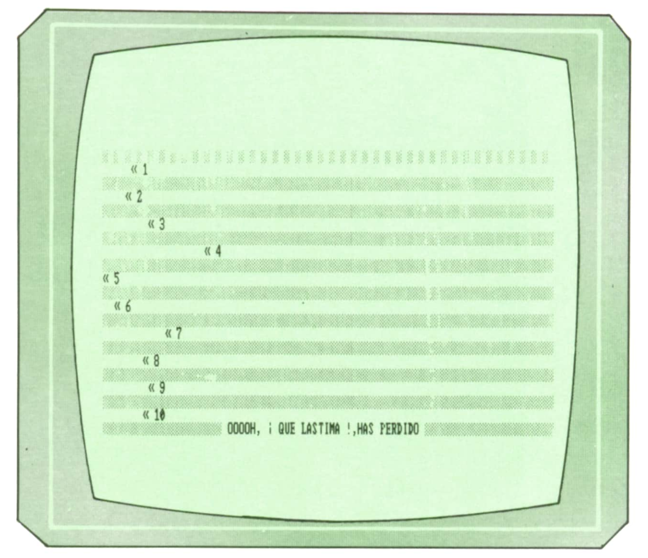



Fig. 2. Preparados para jugar.

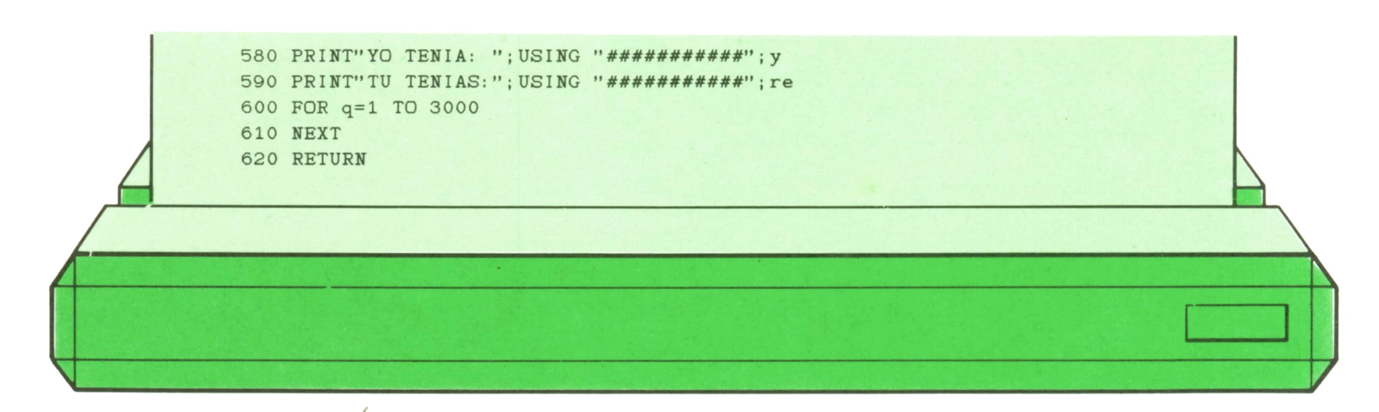
Según el dinero que apuestes, se creará un bote con más o menos dinero. Dinero que te llevarás, junto con lo que apuestes, en caso de que gane tu caballo de carrera.



Programa: Reflejos

El siguiente programa, realizado originariamente en un AMSTRAD, nos permitirá ver qué reflejos tenemos.

```
10 REM *****************************
20 REM ***        REFLEJOS       ***
30 REM *** Un programa realizado ***
40 REM ***          Por          ***
50 REM *** Carlos A. Maria Morin ***
60 REM ***                       ***
70 REM ***     (C) Ediciones     ***
80 REM ***  Siglo Cultural 1987  ***
90 REM *****************************
100 REM
110 MODE 2
120 CLEAR
130 LOCATE 35,1
140 PRINT"REFLEJOS"
150 LOCATE 1,2
160 INPUT "Introduce digitos MAX(10):";nu
170 IF nu>10 THEN 110
180 FOR i=1 TO 10
190 CLS
200 rn=INT(RND(1)*100000+RND(2)*10000)
210 IF rn<500 THEN 200
220 x=(rn+x):zs=100000000+1
230 g=12*x:x=g-INT(g/zs)*zs
240 y=INT(10^nu*x/zs)+1
250 IF y<10^(nu-1) THEN 220
260 SOUND 1,500,15:s=INT(RND*(79))+1
270 w=INT(RND*(24))+1
280 LOCATE s,w
290 PRINT USING "###########";y
300 FOR e=1 TO 70*LEN(STR$(y))
310 NEXT
320 CLS
330 rn=rn+1
340 PRINT i")";
350 INPUT "Respuesta=";re
360 CLS
370 IF re=y THEN GOTO 530
380 IF re<>y THEN GOSUB 580
390 pr=pr+1
400 IF pr=1 THEN PRINT"PRIMER FALLO":FOR q=1 TO 3000:NEXT
410 IF pr=2 THEN PRINT"SEGUNDO FALLO":FOR q=1 TO 3000:NEXT
420 IF pr=3 THEN PRINT"TERCER FALLO":FOR q=1 TO 3000:NEXT
430 IF pr>4 THEN PRINT"SOBREPASA TU FACULTAD MEMORISTICA.":GOTO 490
440 FOR q=1 TO 3000
450 NEXT
460 NEXT i
470 PRINT"PORCENTAJE DE ACIERTOS=";1000*l/100;"%"
480 IF (1000*l/100)<=70 THEN PRINT"ME HAS FALLADO........"
490 PRINT"]Pruebas de nuevo? (S/N)";
500 nu$=INKEY$:IF nu$<>"S" AND nu$<>"s" AND nu$<>"N" AND nu$<>"n" THEN 500
510 IF nu$="S" OR nu$="s" THEN 110
520 END
530 LOCATE 36,12
540 PRINT"**CORRECTO**"
550 SOUND 1,50,15
560 l=l+1
570 GOTO 430
```



```
580 PRINT"YO TENIA: ";USING "###########";y
590 PRINT"TU TENIAS:";USING "###########";re
600 FOR q=1 TO 3000
610 NEXT
620 RETURN
```

El ordenador nos mostrará un número en la pantalla durante un breve espacio de tiempo. A continuación nos preguntará por dicho número. Si somos rápidos podemos adivinarlo.

El número de cifras de dicho número lo podemos definir al principio del programa.

El programa funciona perfectamente en el MSX y IBM con sólo cambiar las siguientes líneas:

110 REM
260 S=INT(RND*79)+1
550 REM

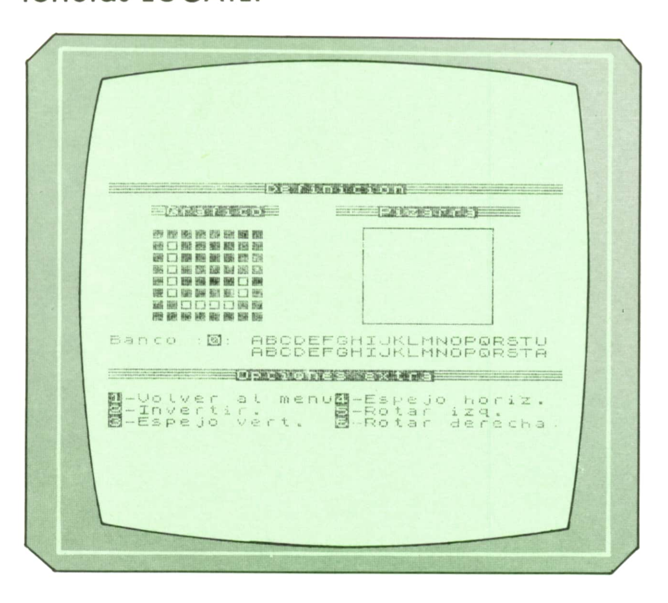
En el caso del IBM hay que cambiar de orden todos los argumentos de las sentencias LOCATE.

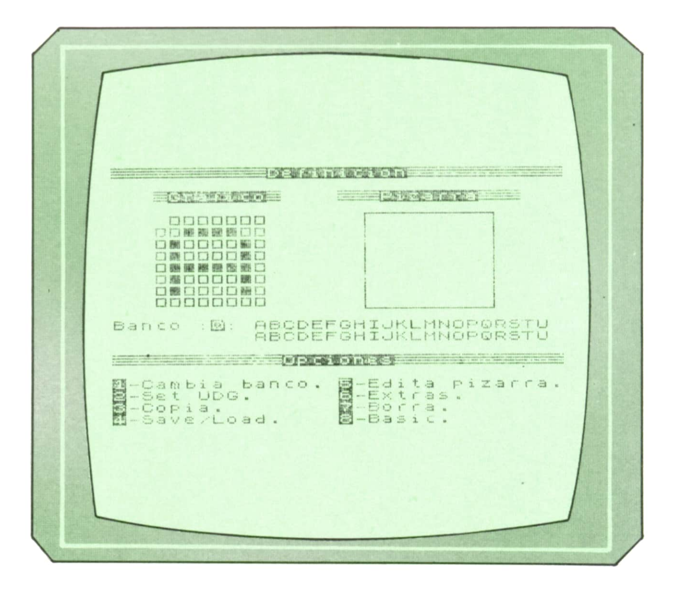


Diseñador de caracteres (UDG) para el SPECTRUM

El siguiente programa que vamos a ver nos permitirá definir nuestros propios caracteres en el SPECTRUM. Poco hay que explicar sobre el programa, ya que es autoexplicativo. Aun así, diremos algunas de sus características.

— Posibilidad de tener 10 bancos distintos de caracteres.
— Inversión de caracteres.
— Rotación y giro de caracteres.
— Pizarra para ver los caracteres definidos.
— Grabación y lectura en y desde casete.






```
PROGRAMA: DISENADOR DE CARACTERES
---------------------------------

 10 REM ************
 20 REM * DISENADOR *
```

```
 30 REM *    D E    *
 40 REM *CARACTERES.*
 50 REM *************
 55 REM
 60 REM ***********************
 70 REM *(c)Ed. Siglo Cultural*
 80 REM *(c)1987.             *
 90 REM ***********************
 95 REM
100 INK 0: PAPER 7: BORDER 7: CLS
110 DEF FN l(A)=A-256*INT (A/256)
120 DEF FN h(A)=INT (A/256)
130 POKE 23658,8
140 REM
145 REM ********************
150 REM * CREADOR DE UDG'S *
160 REM *  INICILIZACION.  *
162 REM ********************
165 REM
170 CLEAR 63687
180 LET DIRBAS=63688
190 LET BANCO=0
200 PRINT AT 11,2; FLASH 1;"POR FAVOR,ESPERE MEDIO MINUTO"
210 FOR g=0 TO 9
220 LET t=g*168+dirbas
230 FOR i=0 TO 167
240 POKE T+i,PEEK (USR "a"+i)
250 NEXT i
260 NEXT g
270 GO SUB 3090
280 RESTORE 330
290 FOR I=USR "A" TO USR "B"+7
300 READ A
310 POKE I,A
320 NEXT I
330 DATA 0,126,66,66,66,66,126,0,0,126,126,126,126,126,126,0
340 CLS
350 LET x1=0
360 LET y1=0
370 LET l=32
380 LET m$="Definicion"
390 GO SUB 2830
400 LET x1=16
410 LET l=13
420 LET y1=2
430 LET m$="Pizarra"
440 GO SUB 2830
450 LET x1=3
460 LET l=9
470 LET m$="Grafico"
480 GO SUB 2830
490 LET y1=16
500 LET x1=0
510 LET l=32
520 LET m$="Opciones"
530 GO SUB 2830
540 DIM g$(8,8)
550 DIM P$(8,9)
560 FOR i=1 TO 8
570 LET g$(i)="AAAAAAAA"
580 NEXT i
590 GO SUB 3130
600 GO SUB 2980
610 PLOT 143,144
620 DRAW 73,0
630 DRAW 0,-65
640 DRAW -73,0
```

```
650 DRAW 0,65
660 GO SUB 3180
670 GO SUB 3250
680 LET c$="A"
690 LET x=3
700 LET y=4
710 GO SUB 3310
711 REM
712 REM ******************
720 REM * BUCLE PRICIPAL *
721 REM ******************
722 REM
730 PRINT AT Y,X;" "
740 PAUSE 12: IF INKEY$<>"" THEN  PRINT AT Y,X;G$(Y-3,X-2): GO TO 770
750 PRINT AT Y,X;G$(Y-3,X-2)
760 PAUSE 12
770 LET A$=INKEY$
780 IF A$<"9" AND A$>"0" THEN  GO TO 840
790 LET X=X+(A$="P" AND X<10)-(A$="O" AND X>3)
800 LET Y=Y+(A$="A" AND Y<11)-(A$="Q" AND Y>4)
810 IF A$<>" " THEN  GO TO 730
820 LET G$(Y-3,X-2)=("A" AND G$(Y-3,X-2)="B")+("B" AND G$(Y-3,X-2)="A")
830 GO TO 730
831 REM
832 REM ************
840 REM * OPCIONES *
842 REM ************
843 REM
850 BEEP .1,24
860 IF A$="1" THEN  GO SUB 970: GO TO 940
870 IF A$="2" THEN  GO SUB 1050: GO TO 940
880 IF A$="3" THEN  GO SUB 1210: GO TO 940
890 IF A$="4" THEN  GO SUB 1270: GO TO 940
900 IF A$="5" THEN  GO SUB 1640: GO TO 940
910 IF A$="6" THEN  GO SUB 2090: GO TO 940
920 IF A$="7" THEN  GO SUB 2650: GO TO 940
930 IF A$="8" THEN  GO SUB 2710
940 INPUT "": PRINT #1;"  O,P,Q,A-Mover 1-8 Opciones"
950 BEEP .1,24
960 GO TO 730
961 REM
962 REM ******************
970 REM * CAMBIO DE BANCO *
972 REM ******************
973 REM
980 INPUT "Seleccione banco (1-9) ";b
990 IF b>9 OR b<0 THEN  GO TO 3470
1000 LET BANCO=B
1010 LET DIRBAS=63688+168*BANCO
1020 GO SUB 2980
1030 GO SUB 3180
1040 RETURN
1041 REM
1042 REM ******************
1050 REM * ASIGNA CARACTER *
1052 REM ******************
1053 REM
1060 INPUT "Caracter (A-U) ? "; LINE b$
1070 IF b$>"U" OR b$<"A" THEN  GO TO 3470
1080 LET c$=b$
1090 GO SUB 3130
1100 FOR g=1 TO 8
1110 LET t=0
1120 LET T2=128
1130 FOR i=1 TO 8
1140 LET t=t+(T2 AND g$(g,I)="B")
1150 LET T2=T2/2
```

```
1160 NEXT i
1170 POKE USR C$+G 1,T
1180 NEXT g
1190 GO SUB 2980
1200 RETURN
1201 REM
1202 REM *********************
1210 REM *COPIA DE UN CARACTER*
1211 REM *A LA PARRILLA.      *
1212 REM *********************
1213 REM
1220 INPUT "Caracter (A-U) ? "; LINE b$
1230 IF B$>"U" OR B$<"A" THEN  GO TO 3470
1240 LET C$=B$
1250 GO SUB 3310
1260 RETURN
1261 REM
1262 REM *************
1270 REM * SAVE/LOAD *
1272 REM *************
1273 REM
1280 INPUT  INVERSE 1;"S"; INVERSE 0;"ave o "; INVERSE 1;"L"; INVERSE 0;"oad ? (
S/L) "; LINE b$
1290 IF b$<>"S" AND b$<>"L" THEN  GO TO 3470
1300 LET x1=0
1310 LET y1=16
1320 LET m$="Save"
1330 LET L=32
1340 IF b$="L" THEN  LET m$="Load"
1350 GO SUB 3510
1360 GO SUB 2830
1370 INPUT "Nombre (Max.10 caracteres)"' LINE n$
1380 IF LEN n$>10 THEN  GO TO 1550
1390 IF B$="L" THEN  GO TO 1490
1391 REM
1392 REM ********
1400 REM * SAVE *
1401 REM ********
1402 REM
1410 IF N$="" THEN  GO TO 1550
1420 SAVE N$CODE DIRBAS,168
1430 INPUT "Desea verificar ? (S/N) "; LINE b$
1440 IF b$="N" THEN  GO TO 1570
1450 PRINT AT 17,0;"Ponga en marcha el cassette"
1460 PRINT AT 17,0;
1470 VERIFY n$CODE dirbas,168
1480 GO TO 1570
1481 REM
1482 REM ********
1490 REM * LOAD *
1492 REM ********
1493 REM
1500 PRINT AT 17,0;"Ponga en marcha el cassette"
1510 LOAD n$CODE dirbas,168
1520 GO SUB 2980
1530 GO SUB 3180
1540 GO TO 1570
1541 REM
1542 REM **********************
1550 REM * ERROR EN EL NOMBRE *
1552 REM **********************
1553 REM
1560 GO SUB 3470
1570 GO SUB 3510
1580 LET X1=0
1590 LET Y1=16
```

```
1600 LET M$="Opciones"
1610 GO SUB 2830
1620 GO SUB 3250
1630 RETURN
1631 REM
1632 REM ***********************
1640 REM *EDICION DE LA PIZARRA*
1642 REM ***********************
1643 REM
1650 GO SUB 3510
1660 LET X1=0
1670 LET Y1=16
1680 LET M$="Edicion de pizarra"
1690 LET l=32
1700 GO SUB 2830
1710 PRINT AT 18,0; INVERSE 1;"5"; INVERSE 0;","; INVERSE 1;"6"; INVERSE 0;",";
INVERSE 1;"7"; INVERSE 0;","; INVERSE 1;"8"; INVERSE 0;"-Movimiento del cursor."
' INVERSE 1;"SPACE"; INVERSE 0;"-Borra el caracter."
1720 PRINT AT 20,0; INVERSE 1;"A"; INVERSE 0;"-"; INVERSE 1;"U"; INVERSE 0;"-Pon
er caracter en la pizarra"
1730 PRINT AT 21,0; INVERSE 1;"ENTER"; INVERSE 0;"-Finaliza edicion."
1740 LET X2=18
1750 LET Y2=4
1751 REM
1752 REM ***********************
1760 REM * BUCLE DE LA EDICION *
1761 REM * DE LA PIZARRA.      *
1762 REM ***********************
1763 REM
1770 FOR I=1 TO 8
1780 FOR G=1 TO 9
1790 LET B$=P$(I,G)
1800 IF B$<>" " THEN  LET B$=CHR$ (CODE B$-79)
1810 LET P$(I,G)=B$
1820 NEXT G
1830 NEXT I
1840 GO SUB 3180
1850 BEEP .1,24
1860 PRINT AT Y2,X2; PAPER 2;" "
1870 PAUSE 10
1880 IF INKEY$<>"" THEN  PRINT AT Y2,X2;P$(Y2-3,X2-17): GO TO 1910
1890 PRINT AT Y2,X2;P$(Y2-3,X2-17)
1900 PAUSE 10
1910 LET A$=INKEY$
1920 IF A$=CHR$ 13 THEN  GO TO 1980
1930 LET X2=X2+(A$="8" AND X2<26)-(A$="5" AND X2>18)
1940 LET Y2=Y2+(A$="6" AND Y2<11)-(A$="7" AND Y2>4)
1950 IF A$<>" " AND (A$<"A" OR A$>"U") THEN  GO TO 1860
1960 LET P$(Y2-3,X2-17)=A$
1970 GO TO 1860
1971 REM
1972 REM ***********************
1980 REM *SALIDA DE LA OPCION 5*
1982 REM ***********************
1983 REM
1990 BEEP .1,24
2000 FOR I=1 TO 8
2010 FOR G=1 TO 9
2020 LET B$=P$(I,G)
2030 IF B$<>" " THEN  LET B$=CHR$ (CODE B$+79)
2040 LET P$(I,G)=B$
2050 NEXT G
2060 NEXT I
2070 GO SUB 3180
2080 GO TO 1570
2081 REM
2082 REM *********************
```

```
2090 REM * OPERACIONES EXTRA *
2092 REM ********************
2093 REM
2100 GO SUB 3510
2110 LET X1=0
2120 LET Y1=16
2130 LET L=32
2140 LET M$="Opciones extra"
2150 GO SUB 2830
2160 PRINT AT 18,0; INVERSE 1;"1"; INVERSE 0;"-Volver al menu"; INVERSE 1;"4"; I
NVERSE 0;"-Espejo horiz."' INVERSE 1;"2"; INVERSE 0;"-Invertir.", INVERSE 1;"5";
 INVERSE 0;"-Rotar izq."
2170 PRINT AT 20,0; INVERSE 1;"3"; INVERSE 0;"-Espejo vert.", INVERSE 1;"6"; INV
ERSE 0;"-Rotar derecha."
2180 LET a$=INKEY$
2190 IF A$>"6" OR A$<"1" THEN  GO TO 2180
2200 BEEP .1,24
2210 IF a$="1" THEN  GO TO 1570
2220 IF a$="2" THEN  GO SUB 2300
2230 IF a$="3" THEN  GO SUB 2380
2240 IF a$="4" THEN  GO SUB 2450
2250 IF a$="5" THEN  GO SUB 2540
2260 IF a$="6" THEN  GO SUB 2590
2270 BEEP .1,24
2280 GO SUB 2920
2290 GO TO 2180
2291 REM
2292 REM ***********************
2300 REM *INVERSION DE CARACTER*
2302 REM ***********************
2303 REM
2310 FOR I=1 TO 8
2320 FOR G=1 TO 8
2330 IF G$(I,G)="A" THEN  LET G$(I,G)="B": GO TO 2350
2340 LET G$(I,G)="A"
2350 NEXT G
2360 NEXT I
2370 RETURN
2371 REM
2372 REM *******************
2380 REM * ESPEJO VERTICAL *
2382 REM *******************
2383 REM
2390 FOR I=1 TO 4
2400 LET B$=G$(I)
2410 LET G$(I)=G$(9-I)
2420 LET G$(9-I)=B$
2430 NEXT I
2440 RETURN
2441 REM
2442 REM *********************
2450 REM * ESPEJO HORIZONTAL *
2452 REM *********************
2453 REM
2460 FOR G=1 TO 8
2470 FOR I=1 TO 4
2480 LET B$=G$(G,I)
2490 LET G$(G,I)=G$(G,9-I)
2500 LET G$(G,9-I)=B$
2510 NEXT I
2520 NEXT G
2530 RETURN
2531 REM
2532 REM *******************
2540 REM * ROTAR IZQUIERDA *
2542 REM *******************
2543 REM
```

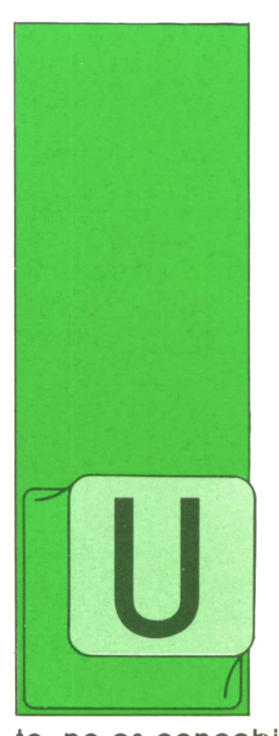
```
2550 FOR I=1 TO 8
2560 LET G$(I)=G$(I,2 TO )+G$(I,1)
2570 NEXT I
2580 RETURN
2581 REM
2582 REM *****************
2590 REM * ROTAR DERECHA *
2592 REM *****************
2593 REM
2600 FOR I=1 TO 8
2610 LET G$(I)=G$(I,8)+G$(I, TO 7)
2620 NEXT I
2630 RETURN
2640 STOP
2641 REM
2642 REM ***********
2650 REM * BORRADO *
2652 REM ***********
2653 REM
2660 FOR I=1 TO 8
2670 LET G$(I)="AAAAAAAA"
2680 NEXT I
2690 GO SUB 2920
2700 RETURN
2701 REM
2702 REM ******************
2710 REM * VUELTA A BASIC *
2712 REM ******************
2713 REM
2720 GO SUB 3510
2730 LET X1=0
2740 LET Y1=16
2750 LET L=32
2760 LET M$="Programa abandonado"
2770 GO SUB 2830
2780 PRINT AT 18,0;"Para volver al programa,teclear        CONTINUE"
2790 STOP
2800 GO TO 1570
2810 LET X=5: LET Y=5
2820 RETURN
2821 REM
2822 REM ***********************
2830 REM *COLOCA ENCABEZAMIENTO*
2832 REM ***********************
2833 REM
2840 FOR i=175-y1*8 TO 175-y1*8-7 STEP -2
2850 PLOT x1*8,i
2860 DRAW l*8-1,0
2870 NEXT i
2880 INVERSE 1
2890 PRINT AT y1,x1+INT ((l-LEN m$)/2);m$
2900 INVERSE 0
2910 RETURN
2911 REM
2912 REM ********************
2920 REM * ESCRIBE PARRILLA *
2922 REM ********************
2923 REM
2940 FOR g=4 TO 11
2950 PRINT AT g,3;g$(g-3)
2960 NEXT g
2970 RETURN
2971 REM
2972 REM **********************
2980 REM * PRINT BANCO EN USO *
2982 REM **********************
2983 REM
```

```
3000 GO SUB 3130
3010 PRINT AT 13,0;"Banco :"; INVERSE 1;banco; INVERSE 0;": ABCDEFGHIJKLMNOPQRST
U"
3020 PRINT AT 14,10;
3030 FOR i=144 TO 164
3040 PRINT CHR$ i;
3050 NEXT i
3060 GO SUB 3180
3070 GO SUB 3090
3080 RETURN
3081 REM
3082 REM **********************
3090 REM *COLOCA EL SET NORMAL*
3092 REM **********************
3093 REM
3100 POKE 23675,88
3110 POKE 23676,255
3120 RETURN
3121 REM
3122 REM ***********************
3130 REM *COLOCA EL SET MARCADO*
3140 REM *POR DIRBAS.          *
3142 REM ***********************
3143 REM
3150 POKE 23675,FN L(DIRBAS)
3160 POKE 23676,FN H(DIRBAS)
3170 RETURN
3171 REM
3172 REM ****************
3180 REM * SACA PIZARRA *
3182 REM ****************
3183 REM
3190 GO SUB 3130
3200 FOR I=1 TO 8
3210 PRINT AT 4+I-1,18;P$(I)
3220 NEXT I
3230 GO SUB 3090
3240 RETURN
3241 REM
3242 REM **************
3250 REM * EDITA MENU *
3252 REM **************
3253 REM
3260 PRINT AT 18,0; INVERSE 1;"1"; INVERSE 0;"-Cambia banco.", INVERSE 1;"5"; IN
VERSE 0;"-Edita pizarra."; INVERSE 1;"2"; INVERSE 0;"-Set UDG.", INVERSE 1;"6";
INVERSE 0;"-Extras."
3270 PRINT AT 20,0; INVERSE 1;"3"; INVERSE 0;"-Copia.", INVERSE 1;"7"; INVERSE 0
;"-Borra.", INVERSE 1;"4"; INVERSE 0;"-Save/Load.", INVERSE 1;"8"; INVERSE 0;"-B
asic."
3280 INPUT ""
3290 PRINT #1;"  O,P,Q,A-Mover 1-8 Opciones"
3300 RETURN
3301 REM
3302 REM *****************
3310 REM * COPIA DEL UDG *
3311 REM * A LA PARRILLA *
3312 REM *****************
3313 REM
3320 GO SUB 3130
3330 DIM G$(8,8)
3340 FOR G=1 TO 8
3350 LET T=PEEK (USR C$+G-1)
3360 LET T2=128
3370 FOR I=1 TO 8
3380 LET g$(g,i)="A"
3390 LET T3=T-T2
3400 IF T3>=0 THEN  LET T=T3: LET g$(g,I)="B"
```

```
3410 LET T2=T2/2
3420 NEXT I
3430 NEXT G
3440 GO SUB 3090
3450 GO SUB 2920
3460 RETURN
3461 REM
3462 REM **********************
3470 REM *ERROR EN UNA ENTRADA*
3472 REM **********************
3473 REM
3480 BEEP 1,0: PRINT #1;TAB 7; FLASH 1;"ERROR:OPCION ANULADA"
3490 PAUSE 200
3500 RETURN
3501 REM
3502 REM ***********************
3510 REM * BORRADO DE LA PARTE *
3511 REM * DESTINADA AL MENU.  *
3512 REM ***********************
3513 REM
3520 FOR I=16 TO 21
3530 PRINT AT I,0;"
3540 NEXT I
3550 INPUT ""
3560 RETURN
```

TECNICAS DE ANALISIS

BASES DE DATOS



NA de las "herramientas" más comúnmente utilizadas hoy en día en las aplicaciones informáticas son los "sistemas de gestión de bases de datos" (SGBD). En efecto, no es concebible ya un sistema informático de envergadura media o grande en que se gestionen las informaciones mediante comandos dados directamente en el lenguaje de programación en que se haya preparado el resto de los procesos.

Están disponibles los grandes SGBD's, de los que hablaremos a continuación, pero también hay pequeños (aunque no por ello menos eficaces) sistemas de manejo de datos a nivel de los ordenadores personales: DBASE II y III, PFS, SYMPHONY, FRAMEWORK, etc.

Tal como define C. Chrisment, una base de datos es "un conjunto estructurado de informaciones, agregadas o elementales, accesible por un grupo de utilizadores". Los objetivos fundamentales que se pretenden conseguir con la implementación de una base de datos, objetivos vitales en un gran sistema informático pero, también, útiles en cualquier sistema personal o profesional, son:

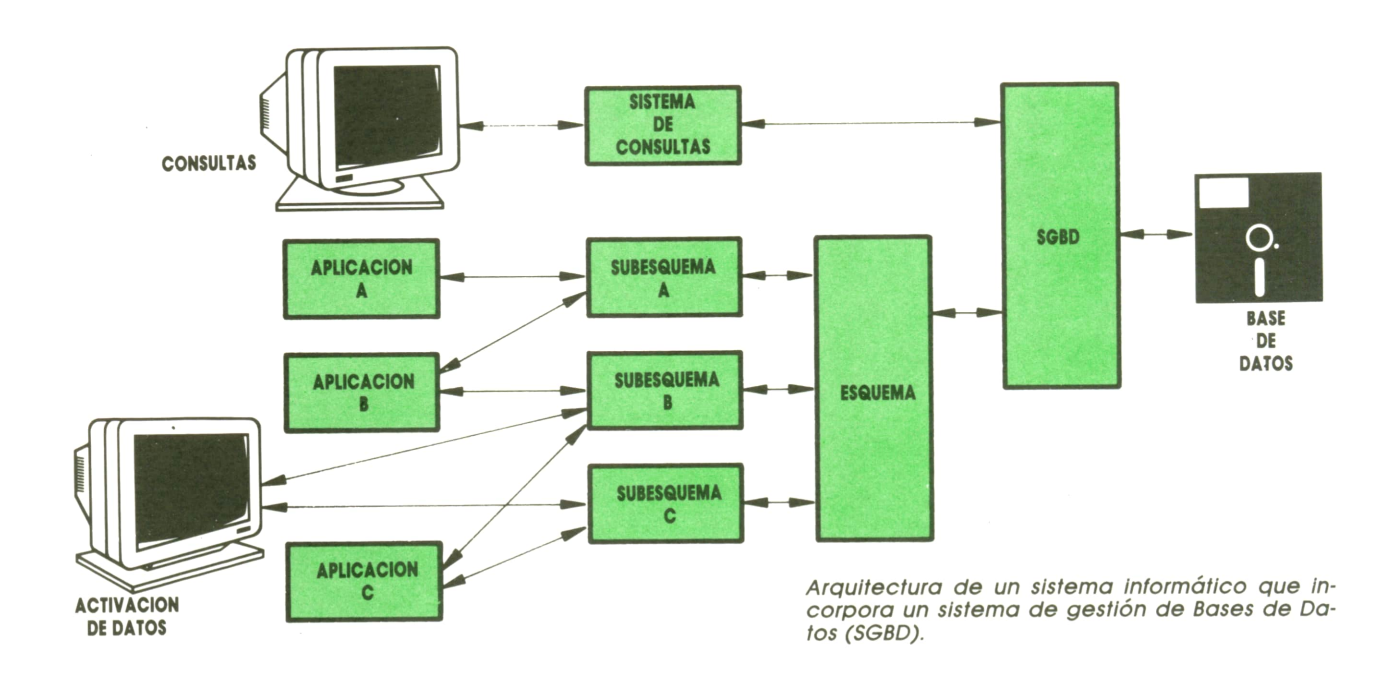
A. *Centralizar y unificar la información.* Con esta centralización se suprime la redundancia de informaciones (cada dato está almacenado solamente una vez en el sistema); la búsqueda de una información se realiza en un solo sitio (menos tiempo de búsqueda y más seguridad en la localización de los datos); además, se centralizan los controles, las actualizaciones periódicas, los procesos comunes, etc.

B. *Asegurar la independencia entre datos y procedimientos.* Es una ventaja específica de los sistemas de gestión de bases de datos: por definición, todo sistema de este tipo debe contar con herramientas adecuadas para la definición de los datos y su estructura, independientemente de los recursos que aporte para el manejo posterior de estos datos (es el propio SGBD quien se encarga de definir y mantener los procedimientos que aseguren la relación entre los procesos externos que deben acceder a los datos y los propios datos). Esta misma independencia y la propia concepción del SGBD permiten, si es necesario posteriormente, reestructurar cómodamente las informaciones ya almacenadas en la base de datos.

C. *Relacionar datos de distintos procesos.* Uno de los problemas cotidianos del analista es definir, controlar y asegurar el adecuado mantenimiento de los datos comunes a varias aplicaciones. Si se implementa una base de datos común accesible desde varios procesos, la integridad de los datos, su coherencia, su estabilidad, etc., están asegurados por el propio SGBD, sin intervención adicional del analista.

D. *Asegurar la integridad de los datos.* Tanto por la cualidad ya reseñada de unificar los datos como por los propios sistemas automáticos de manejo de las informaciones, los SGBD's aportan una mayor garantía de que, en el transcurso del tiempo, no se va a a deteriorar la integridad de los datos que se mantienen en la base de datos.

E. *Aportar la necesaria confidencialidad.* No es una cualidad intrínseca a los SGBD's, pero la mayoría de ellos aportan una serie de controles de confidenciali-



Arquitectura de un sistema informático que incorpora un sistema de gestión de Bases de Datos (SGBD).

dad (básicamente bajo la fórmula de palabras clave) que garantizan una mayor seguridad en el manejo de informaciones confidenciales, tanto en entornos de funcionamiento monousuario como en entornos multiusuario.

F. *Fiabilidad en la "compartición" de datos.* Un elemento clave de cualquier SGBD está formado por los mecanismos que aseguran la adecuada gestión de los datos, en entornos complejos: existen sistemas que controlan los conflictos que pueden surgir cuando dos procesos distintos intentan acceder a un mismo dato, se suelen aceptar operaciones bajo la forma de "transacciones" complejas (en una "transacción" se agrupan varias operaciones que deben ir ligadas, de tal modo que sólo son efectivos los cambios realizados en la base de datos cuando se ha concluido la transacción completa, evitando de este modo actualizaciones parciales —erróneas— de la base de datos), y otros mecanismos complejos de gestión de datos.

 El administrador de la base de datos

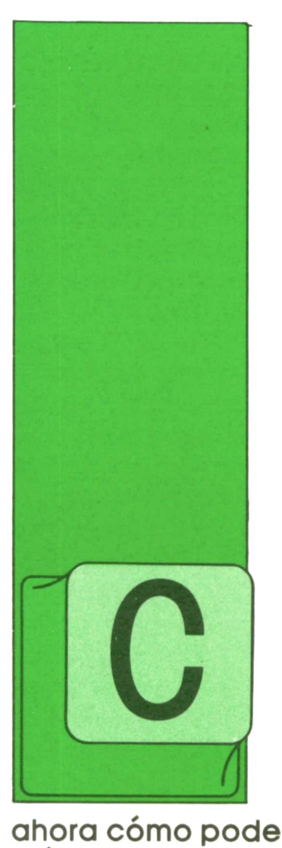
Aunque el SGBD produce de un modo automático la gestión de todas las informaciones involucradas en el sistema, es necesario mantener una coordinación entre todos los usuarios y hay que definir las estructuras subyacentes en la base y los procesos generales de mantenimiento y/o actualización del sistema. Todas estas tareas suelen ser realizadas por una misma y única persona, experta en el SGBD de que se trate y en los lenguajes de definición y manipulación de datos correspondientes, que se suele llamar "administrador" de la base de datos. Son tareas específicas del administrador de la base las siguientes:

— diseñar el esquema de la base de datos y los subesquemas necesarios;
— decidir las técnicas de acceso a la base;
— decidir los sistemas concretos de implantación física;
— establecer la relación del SGBD con utilizador (o utilizadores);
— definir una estrategia de relanzamiento, en caso de que se produzca algún incidente;
— medir las características del sistema (capacidades, tiempos de acceso, volúmenes, etc.) y decidir modificaciones si son necesarias.

Naturalmente, para realizar estas tareas el administrador de la base debe ser un experto en el SGBD que se esté utilizando, por lo que suele actuar de soporte técnico a los utilizadores del sistema.

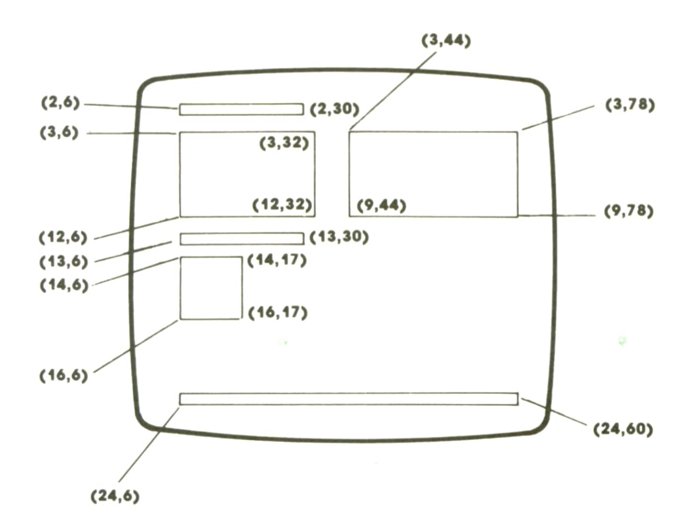
TECNICAS DE PROGRAMACION

Manejo de la pantalla (continuación)

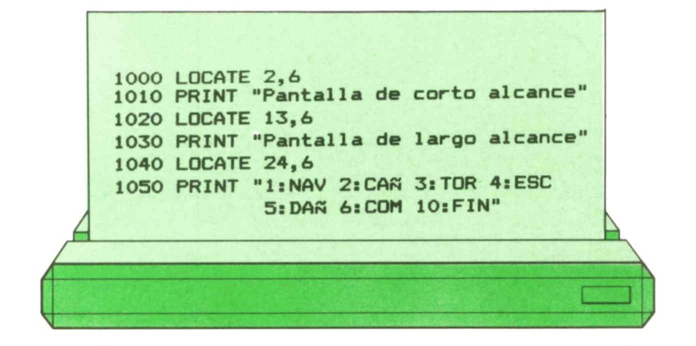


ONTINUANDO con el ejemplo que comenzamos en el capítulo anterior (la preparación de una pantalla adecuada para un juego de guerra espacial), vamos a ver ahora cómo podemos escribir la información que debe aparecer en cada uno de los campos de la pantalla y cómo podemos conseguir que el programa sepa cuál de las teclas ha sido presionada para realizar una acción determinada.

En primer lugar, recordemos la distribución de los campos en la pantalla:



Tres de los campos sólo tienen una línea: aquellos cuya esquina superior izquierda se encuentra en los puntos (2,6), (13,6) y (24,6) (recuérdese que la posición de un punto en la pantalla queda definida por el número de la fila y la columna donde se encuentra). Por tanto, escribir en cada uno de ellos la información deseada es muy fácil. Basta con colocar el cursor en la posición correspondiente (la posición inicial del campo) y escribir el texto deseado utilizando la función PRINT:



Rellenar un campo de varias líneas es sólo un poco más complicado, pues hay que escribir independientemente cada una de las líneas y colocarse al principio de cada una con su correspondiente instrucción LOCATE.

Por ejemplo, supongamos que lo que se ve en la pantalla de corto alcance lo tenemos en la variable Q$, que ha sido definida previamente como una tabla o matriz de 8 filas y 27 columnas. Esta información vamos a presentarla en el campo cuya esquina superior izquierda se encuentra en el punto (3,6). Este campo tiene, sin embargo, 10 filas. La primera y la última están reservadas para escribir dos líneas horizontales que enmarquen la pantalla de corto alcance.

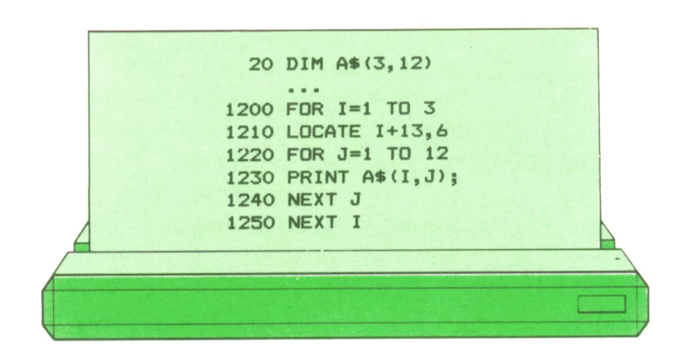
Para escribir toda esta información, llenar el campo y construir la pantalla de corto alcance, podremos utilizar el siguiente programa:

```
  10 DIM Q$(8,27)
     ...
1100 LOCATE 3,6
1110 PRINT "___________________________"
1120 FOR I=1 TO 8
1130 LOCATE I+3,6
1140 FOR J=1 TO 27
1150 PRINT Q$(I,J);
1160 NEXT J
1170 NEXT I
1180 LOCATE 12,6
1190 PRINT "___________________________"
```

Observemos este programa: en la línea 10 se ha definido la variable Q$ como una tabla o matriz de 8 filas y 27 columnas. En alguna parte intermedia del programa (representada aquí por los puntos suspensivos) se habrán asignado a esta variable los valores adecuados (caracteres, pues Q$ es una variable de tipo literal). Ahora, en la instrucción de etiqueta 1100, ha llegado el momento de escribirlos en la pantalla. Esto se consigue realizando los siguientes pasos:

1. La primera línea del campo (que ha de rellenarse con una línea horizontal) se escribe exactamente igual que en los ejemplos anteriores: hace falta una instrucción LOCATE para colocar el cursor en el punto deseado, y una instrucción PRINT, para escribir el mensaje correspondiente.
2. Ahora escribimos las líneas intermedias, cada una de las cuales corresponde a una fila de la matriz. Por tanto, tenemos que formar un bucle de programa que vaya recorriendo, una a una, dichas filas. Este bucle está comprendido entre las instrucciones 1120 y 1170. La variable que controla este primer bucle y señala sucesivamente a cada fila se llama I en el ejemplo.
3. Al principio de cada fila tenemos que colocar el cursor en la línea de la pantalla correspondiente. Obsérvese que la primera fila debe escribirse a partir del punto (4,6); la segunda, a partir del punto (5,6); y así sucesivamente. Por tanto, si I es igual al número de la fila que Q$ que vamos a escribir, el cursor deberá colocarse en el punto (I+3,6). Esto es lo que conseguimos con la línea 1130.
4. Ahora hay que escribir una fila entera de Q$ utilizando la función PRINT. Lo que haremos será construir un nuevo bucle y escribirla carácter a carácter. Este bucle (que abarca desde la instrucción 1140 hasta la 1160) está controlado por la variable J, que, naturalmente, varía de 1 a 27 (pues Q$ tiene 27 columnas).
5. La instrucción PRINT, de etiqueta 1150, se ejecuta, por tanto, 27 veces a medida que vamos escribiendo en la pantalla una línea de la matriz. Obsérvese que, para evitar que el cursor se nos mueva al principio de la línea siguiente de la pantalla, hemos terminado esta instrucción con un punto y coma. Con ello conseguimos que los 27 caracteres vayan apareciendo sucesivamente, unos al lado de otros, en la posición deseada.
6. Cuando termina la ejecución del segundo bucle (porque J ha alcanzado el valor 27), debe continuar la ejecución del primero (es decir, el valor de I se incrementa en una unidad). Entonces, si todavía no hemos alcanzado el valor 8, se vuelve a repetir todo el proceso, escribiéndose una nueva línea. En total, la instrucción 1150 se ejecuta 8 × 27 = 216 veces, una por cada carácter de la matriz Q$.
7. Finalmente, una vez terminados los dos bucles, escribimos en la pantalla la segunda línea horizontal, que cierra el campo por debajo. Para ello necesitaremos de nuevo una instrucción LOCATE y otra PRINT.

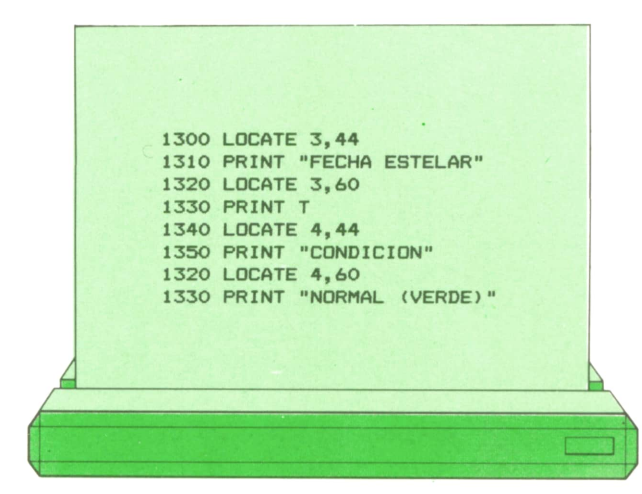
De igual manera procederemos con la pantalla de largo alcance, cuya información suponemos que se encuentra en la variable A$, que es una matriz de 3 filas y 12 columnas. Veamos el programa correspondiente, que no vamos a detallar:



Esta vez no hemos dibujado dos rayas horizontales enmarcando el campo.

Finalmente, veamos el campo donde aparece el informe condensado de la si-

tuación de la nave. Vamos a considerar únicamente dos líneas de este campo (las demás son análogas). La primera línea presentará la fecha estelar en que nos encontramos, cuyo valor tendremos en la variable T. La segunda presentará la condición general de la nave que, dependiendo de las circunstancias, puede ser una de las siguientes: «NORMAL», «ALERTA ROJA», «ALERTA AMARILLA» o «EN PUERTO». Supondremos que, en este momento, la condición es «NORMAL». Veamos la parte del programa que escribiría estas dos líneas:



Obsérvese que cada línea de este campo la hemos pintado en dos veces. De esta forma conseguimos que los valores de las variables que indican la situación de la nave comiencen todos en la misma columna, lo que facilita la lectura y mejora el aspecto de la pantalla.

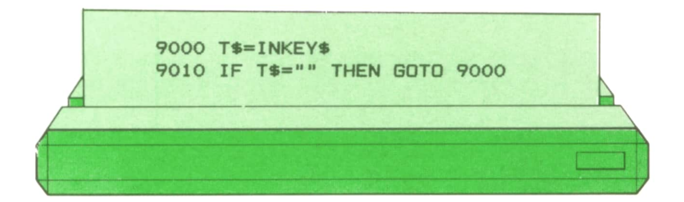
Veamos cómo quedará la pantalla después de pasar el programa completo:

Sólo falta saber cómo puede el programa darse cuenta de que el jugador ha presionado una tecla determinada. Para ello utilizaremos otra instrucción BASIC:

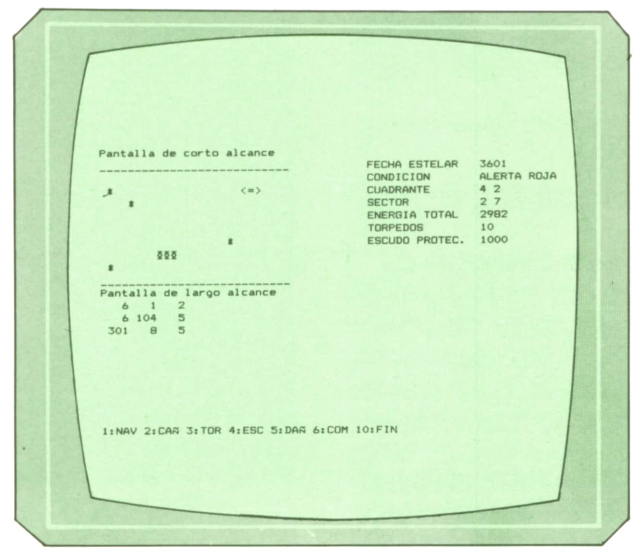
A$=INKEY$

que comprueba si la persona sentada al terminal ha presionado recientemente una tecla. En caso afirmativo, asigna a la variable A$ el carácter correspondiente a dicha tecla. En caso negativo, le asigna una cadena de caracteres vacía (una cadena de cero caracteres). El carácter leído no se imprime en la pantalla, al revés de lo que ocurre automáticamente en la instrucción INPUT. (El nombre A$ puede variar, pero siempre deberá tratarse de una variable de tipo literal.)

El programa siguiente utiliza la instrucción INKEY$ para esperar hasta que se presione una tecla, proporcionándonos entonces el carácter correspondiente a la tecla presionada en la variable T$.



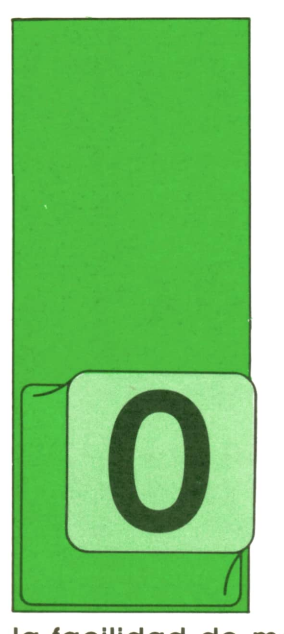
Se verá que la instrucción 9010 devuelve de nuevo control a la instrucción 9000 (y realiza una nueva operación INKEY$) si el resultado obtenido fue la cadena de caracteres vacía (representada por dos dobles comillas seguidas). Es decir, mientras la persona sentada al terminal no presione alguna tecla, la operación INKEY$ se repetirá continuamente.



APLICACIONES



Open Access



PEN ACCESS constituye el más famoso paquete integrado que actualmente se puede encontrar en el mercado. La gran integración de todos sus módulos, incluida la facilidad de manejo y su gran potencia, hacen de este paquete una de las más poderosas herramientas para ordenadores personales.

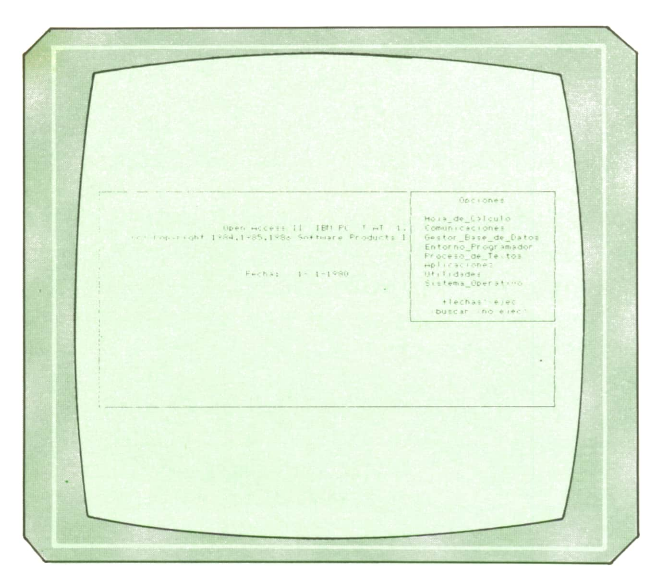




Fig. 1. Pantalla de opciones de Open Access.

Este paquete está compuesto por seis módulos:

— Base de datos.
— Hoja electrónica.
— Procesador de textos.
— Agenda.
— Comunicaciones.
— Gráficos.

Que, sin duda, resultarán más que suficientes para la mayoría de los usuarios.

El ordenador requiere un número de

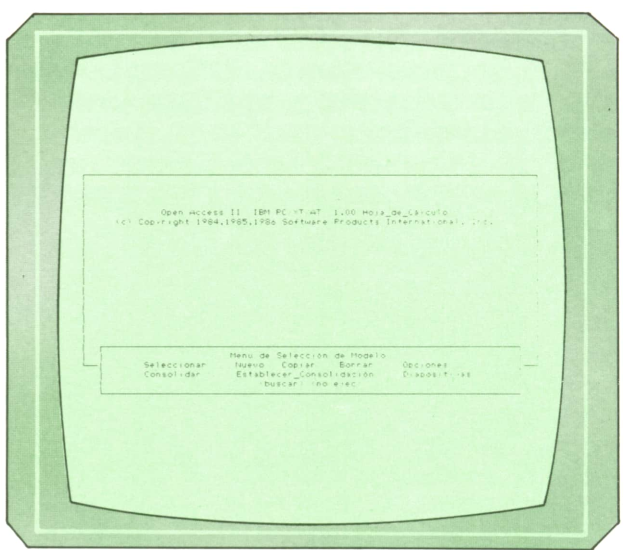




Fig. 2. Opciones de la hoja de cálculo.

192 K de memoria, y aunque funcionara sin problema con dos unidades de disco, es recomendable instalar un disco duro, para evitar, de esta forma, la introducción repetida de disketes. En cuanto a las impresoras, el programa permite soportar varios modelos distintos, tales como: EPSON, C-ITOH, NEC, etc.



Base de datos

A diferencia de otros programas en el que la hoja de cálculo constituye el módulo central, en OPEN ACCESS este módulo lo constituye la base de datos.

Esta base de datos es de tipo relacional, permitiendo el manejo simultáneo de hasta cinco ficheros, en los que el máximo número de registros es de 32.000, con 56 campos cada uno.

La recuperación de datos de los ficheros se realiza a través de un lenguaje que incluye únicamente cuatro comandos: DE, ELIGE, CUYO y ORDEN, que resultan más que suficientes en la mayoría de las

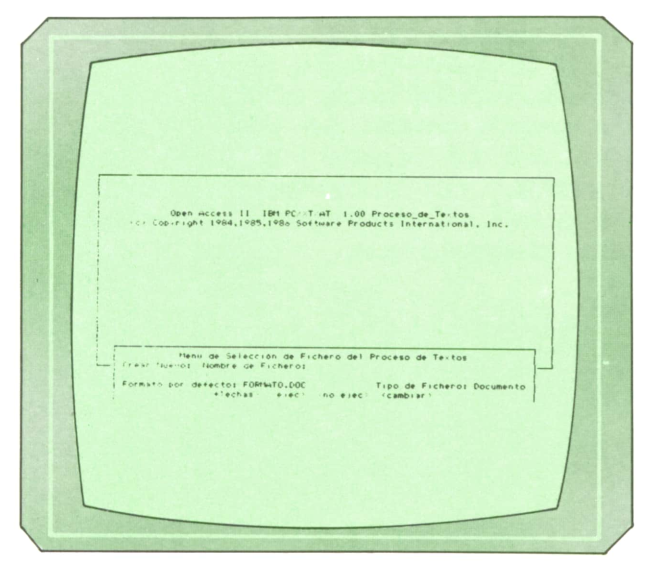




Fig. 3. Pantalla de ayuda.

aplicaciones. Este módulo permite manejar hasta 15 campos clave por fichero. El lenguaje que utiliza la base de datos permite la creación fácil de pantallas para la introducción de datos, así como la generación de informes.

Hoja de cálculo

La hoja de cálculo de OPEN ACCESS permite un manejo de 3.000 filas y 216 columnas.

Una de las ventajas de la hoja electrónica es la rapidez con que efectúa sus cálculos, ya que utiliza la técnica de la memoria virtual, procedimiento que consigue una rápida ejecución de los cálculos, unido a una gran capacidad de almacenamiento.

La hoja electrónica de OPEN ACCESS permite presentar hasta seis ventanas interactivas, de tal forma que es posible relacionar los distintos modelos de hojas de cálculo entre sí.

Además, dispone de las funciones típicas de una hoja de cálculo, como son: funciones matemáticas, estadísticas, lógicas y otras; permite la persecución de objetivos, en los que se conoce el resultado final, pero se desconoce alguno de los datos que conducen a él; OPEN ACCESS se encarga entonces de calcular las incógnitas.

La comunicación con los otros módulos es total, excepto con la agenda, con la que no es posible comunicarse directamente.



Gráficos

OPEN ACCESS permite la confección de gráficos tanto bidimensionales como tridimensionales, tanto de barras, líneas o pastel. Este módulo toma los datos de la hoja electrónica, del procesador de textos, de la base de datos y del módulo de comunicaciones. Este módulo gráfico incluye la posibilidad de realizar gráficos independientes, como si fueran dispositivos, que pueden ser llamados directamente y mandados a otros módulos del programa. En cada gráfico se pueden cambiar las texturas, el color o la perspectiva, permitiendo incluir cabeceras, pies y títulos.



Agenda

La agenda constituye un complemento a los otros módulos del programa, permitiendo la planificación de citas y compromisos, así como el manejo de un fichero con direcciones y teléfonos consultables en cualquier momento. Este módulo incluye un calendario en el que se pueden incluir notas, y que se genera analógicamente o a partir de la ficha del sistema.

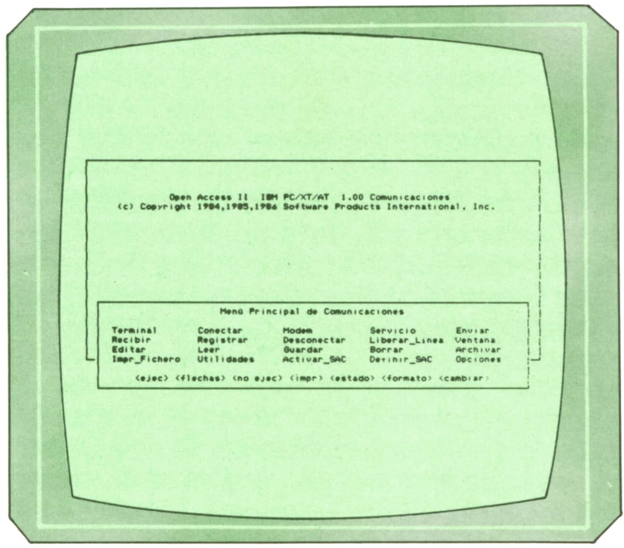




Fig. 4. Opciones de comunicaciones.

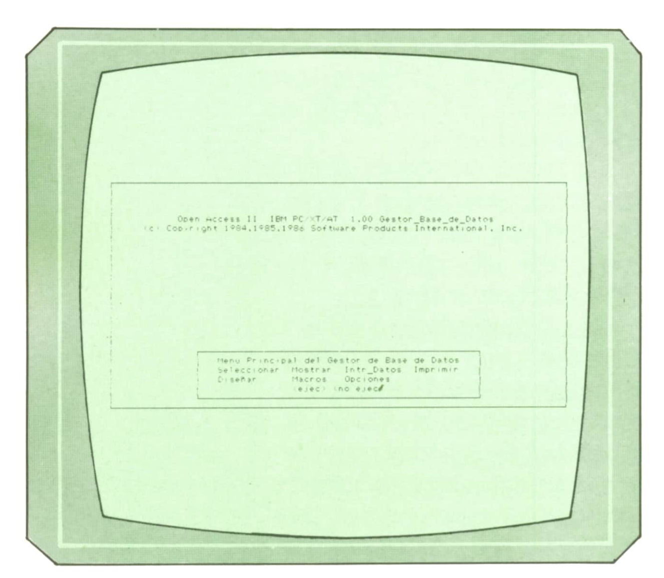




Fig. 5. Opciones de base de datos.

 Comunicaciones

Este módulo es el encargado de comunicar el OPEN ACCESS con otros programas y ordenadores, utilizando para ello la salida RS-232, permitiendo seleccionar la velocidad de transmisión y generar un directorio telefónico, así como realizar llamadas de forma automática.

 Tratamiento de textos

El procesador de textos ofrecidos por OPEN ACCESS se encuadra dentro de la norma, es decir, ofrece lo que puede dar cualquier otro programa de estas características.

Dispone de todas las funciones normales de insertado, borrado, movimiento de bloques, copias, etc.

El usuario puede moverse por el texto con las teclas de control del cursor, definir el formato de texto e incluso incluir textos acabados en el que tenga en curso.

Otras características del referido tratamiento de textos es la posibilidad de crear abreviaturas, 10 en total, función útil cuando se ha de repetir una palabra numerosas veces en un mismo texto. También se pueden definir tipos de párrafos y confeccionar el texto conforme a la estructura creada (máximo de ocho definiciones).

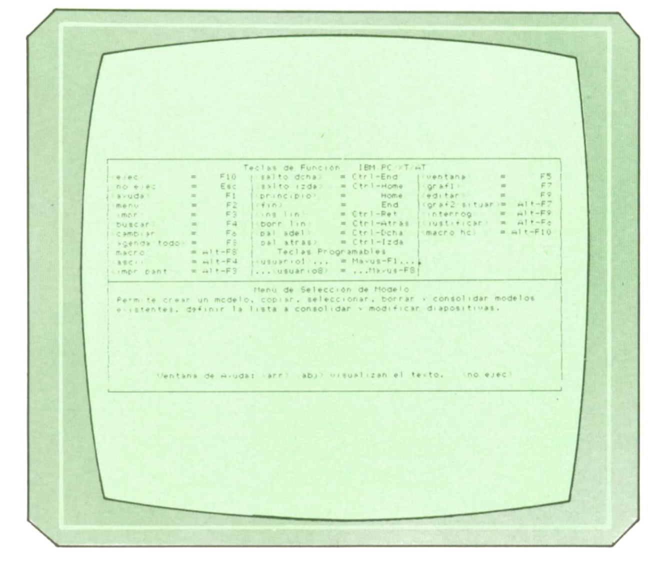




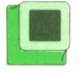
Fig. 6. Opciones del procesado de textos. Modelo de la memoria de un ordenador.

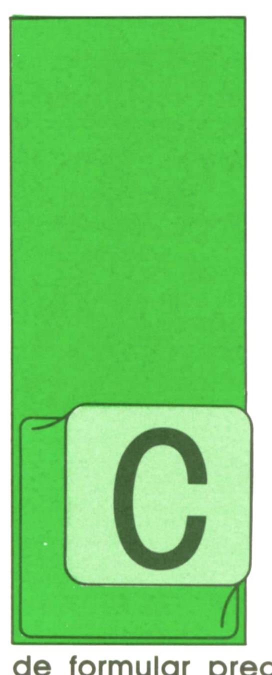
La impresión ofrece las posibilidades comunes de cabeceras, pies, subcabeceras, determinación de la longitud de página. Acepta la inclusión de gráficos en el documento y distingue atributos relativos al tipo de letra, como subrayado, itálica, negrilla.

Siguiendo la tónica general del paquete, el módulo de tratamiento de textos también utiliza la técnica de ventanas para mostrar los comandos y otras funciones.

Está presente asimismo la posibilidad de dividir la pantalla para visualizar dos textos simultáneamente. Este módulo puede comunicarse con cualquiera de los módulos restantes, excepto la agenda electrónica.

PASCAL

 Un programa experto



COMO último ejemplo de aplicación de las estructuras árbol, vamos a desarrollar un programa que, manejando una base de datos, sea capaz, por una parte, de, a base de formular preguntas, dar con la respuesta a un determinado problema y, por otra parte, de ir ampliando la información de esa base de datos a partir de las respuestas obtenidas.

Los programas capaces de manejar el conjunto de conocimientos de un experto en un determinado área del saber humano para ponerlo de manera eficaz al alcance de otras personas, se denominan "programas expertos" y constituyen uno de los aspectos más importantes de lo que se ha dado en llamar "inteligencia artificial".

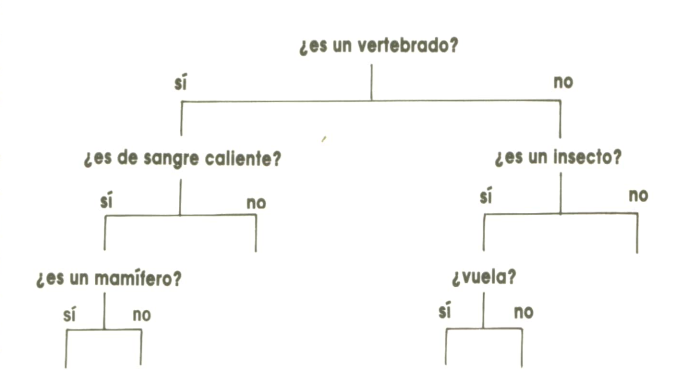
El programa que vamos a desarrollar, de una gran sencillez, se encuentra a muchísima distancia de lo que se entiende hoy día por programa experto, pero, no obstante, puede servir para dar una primera idea de en qué consiste la inteligencia artificial.

En concreto, nuestro programa va a ser capaz de ir preguntando diferentes características de animales hasta dar con uno que se ajuste a todas las respuestas que le hayamos facilitado. Cuando el animal deducido no sea el que debiera, el programa intentará obtener nueva información. Empecemos a construirlo.

Supongamos que la primera vez que corremos el programa éste carezca de datos: no sabría qué preguntar; por ello, habrá que poner ya de entrada en el programa un mínimo de información. Por ejemplo, la primera vez el programa preguntará directamente por un animal en concreto, digamos, la mosca. Si acertara, ahí acabaría todo. Sin embargo, imaginemos que el animal que habíamos pensado fuera el perro.

Al contestar al programa que no era la mosca, éste nos debería preguntar qué animal constituye la solución (el perro) y alguna propiedad que lo distinga de la mosca (ladra). Con esta nueva información, al siguiente intento de descubrir un animal, el programa debería preguntar primero si ladra, para, según fuera la respuesta, preguntar a continuación por el perro o por la mosca. De esta manera el ordenador iría "aprendiendo" nuevos animales y nuevas preguntas que hacer antes de preguntar por uno en concreto.

Supongamos que, en un momento dado, la primera pregunta de todas fuese "¿es un vertebrado?". Si la respuesta a ella fuera SI, habría que hacer a continuación alguna pregunta lógica para vertebrados, por ejemplo "¿es de sangre caliente?", mientras que si hubiera sido NO, la siguiente pregunta debería ser distinta. Para saber la secuencia de preguntas a hacer podríamos utilizar un esquema como el siguiente:



y así hasta llegar a preguntar por un animal en concreto.

Salta a la vista que la estructura resultante es de tipo árbol; para llevarla a la práctica bastaría con utilizar registros con un campo para guardar la pregunta y otros dos de tipo puntero para indicar

a qué registros se debe acudir según la respuesta. Tras los nodos que preguntan por animales concretos no habría más registros. El programa empezaría así:

```
program Animales;

const
 LongPreg = 50;   (* máximo número de caracteres por pregunta *)

type
 Preg_t   = array [1..LongPreg] of char; (* para las preguntas *)
 Punt_t   = ^Ficha_t;
 Ficha_t  = record
             Pregunta: Preg_t;
             QueSi,QueNo: Punt_t
            end;
```

Veamos ahora el procedimiento para determinar la secuencia de preguntas a partir de un nodo dado:

"Mirar en árbol desde el nodo tal:"

¿Es un nodo final?

SI: Preguntar por el animal que contiene.

— Si se ha acertado, se acabó.

— Si se ha fallado, preguntar en qué animal se ha pensado y sus propiedades, ampliar con ello el árbol y terminar.

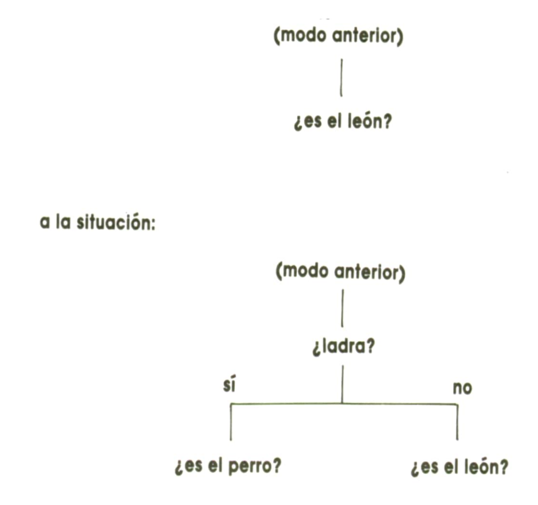
NO: Formular la pregunta que contiene.

— Si la respuesta es SI, mirar en árbol desde el nodo indicado por el puntero QueSi.

— Si la respuesta es NO, mirar en árbol desde el nodo indicado por el puntero QueNo.

Como se ve, el procedimiento es recursivo, pues se llama a sí mismo. Para empezar una búsqueda se llamaría al procedimiento desde el programa principal para "mirar en árbol desde el nodo raíz".

Veamos ahora cómo ampliar el árbol con nueva información. Supongamos que se ha llegado a un nodo final que contiene "el león" y que la respuesta ha sido NO. Si el animal resultara ser "el perro" y la propiedad que lo distingue "ladra", deberíamos pasar de la situación:



o sea, antes de preguntar por el león, preguntar si ladra por si acaso es el perro. El procedimiento sería:

"Ampliar el árbol en el nodo tal:"

1. Reservar sitio para dos nuevos registros a los que apunten QueSi y QueNo.
2. Guardar en el primero de ellos el nuevo animal y hacer sus dos punteros iguales a NIL (es un nodo final).
3. Guardar en el segundo el animal por el que se preguntó y hacer sus dos punteros iguales a NIL.
4. Guardar la propiedad distintiva en el nodo en cuestión.

El programa definitivo queda así:

```
program Animales;

const
 LongPreg = 50;     (* máximo número de caracteres por pregunta *)
 (* si se cambia, retóquese también el principio del programa  *)

type
 Preg_t   = array [1..LongPreg] of char; (* para las preguntas *)
 Punt_t   = ^Ficha_t;
 Ficha_t  = record
             Pregunta: Preg_t;
             QueSi,QueNo: Punt_t
            end;

var
 Raiz: ^Ficha_t; (* Sirve para apuntar al nodo raiz *)

(*------------------------------------------------------------*)
procedure PlanteaPregunta (T: Preg_t);
  (*------------------------------------------------*)
  (* Escribe la pregunta quitando lo que sobra por la *)
  (* derecha para que la interrogación quede pegada.  *)
  (*------------------------------------------------*)

   (*----------------------------------------------------*)
   function  Long: integer;
    (*--------------------------------*)
    (* Devuelve la longitud real de T *)
    (*--------------------------------*)
     var I: integer; Parar: boolean;
    begin
     I:= LongPreg;
     Parar:= false;
     while (I > 0) and not Parar do
       if (T[I] <> ' ') and (T[I] <> chr(0)) then Parar:= true
                                            else I    := I-1;
     Long:= I
    end;
   (*----------------------------------------------------*)

    var I: integer;
  begin
   for I:= 1 to Long do write (T[I])
  end;

(*------------------------------------------------------------*)
function  Afirmativo: boolean;
  (*----------------------------------------------*)
  (* Lee respuesta y devuelve TRUE si ha sido SI *)
  (*----------------------------------------------*)

    var C: char;
  begin
   write ('? (S/N) ');
   readln (C);
   Afirmativo := (C <> 'n') and (C <> 'N')
  end;

(*------------------------------------------------------------*)
procedure Ampliar (Q: Punt_t);
  (*---------------------------------------------*)
  (* Amplia el árbol en el nodo apuntado por Q *)
  (*---------------------------------------------*)

  var Copia: Ficha_t;
 begin
  with Q^ do
   begin
    new (QueSi);
    new (QueNo);
    writeln;
    write ('¿Qué animal es? (con "el" o "la" por delante): ');
    readln (QueSi^.Pregunta);
    QueSi^.QueSi:= nil;
    QueSi^.QueNo:= nil;

    QueNo^.Pregunta:= Pregunta;
    QueNo^.QueSi    := nil;
    QueNo^.QueNo    := nil;

    repeat
     writeln ('Déme una propiedad que lo distinga:');
```

```
     readln (Pregunta);
     writeln ('Entonces, si pregunto:');
     write ('¿'); PlanteaPregunta (Pregunta); writeln ('?');
     write ('y la respuesta es SI, ¿puedo suponer que es ');
     PlanteaPregunta (QueSi^.Pregunta)
    until Afirmativo;

    writeln;
    writeln ('Gracias. Hasta otra.')
   end
end;

(*-----------------------------------------------------------------*)
procedure BuscaEn (P: Punt_t);
  (*----------------------------------------------------*)
  (* Avanza por el árbol desde el nodo apuntado por P *)
  (*----------------------------------------------------*)

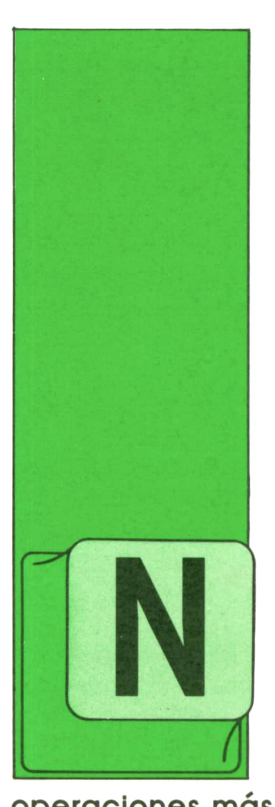
 var C: char;
begin
 with P^ do
  if (QueSi = nil) and (QueNo = nil) then   (* si es final: *)
      begin
       write ('¿es ');
       PlanteaPregunta (Pregunta);
       if  not Afirmativo  then  Ampliar (P)
             else begin writeln; writeln ('Vale, hasta otra.') end;
      end
     else
      begin
       write ('¿');
       PlanteaPregunta (Pregunta);
       if Afirmativo then BuscaEn (QueSi)
                     else BuscaEn (QueNo)
      end
end;
(*-----------------------------------------------------------------*)

begin
(* Al principio sólo hay un nodo que por fuerza ha de ser final: *)
 new (Raiz);
 with Raiz^ do
  begin
   Pregunta:='la mosca
   QueSi    := nil;
   QueNo    := nil
  end;

 repeat
  writeln;
  BuscaEn (Raiz);
  writeln;
  write ('¿Desea seguir');
 until not Afirmativo
end.
```

OTROS LENGUAJES

MODULA-2 (IV)





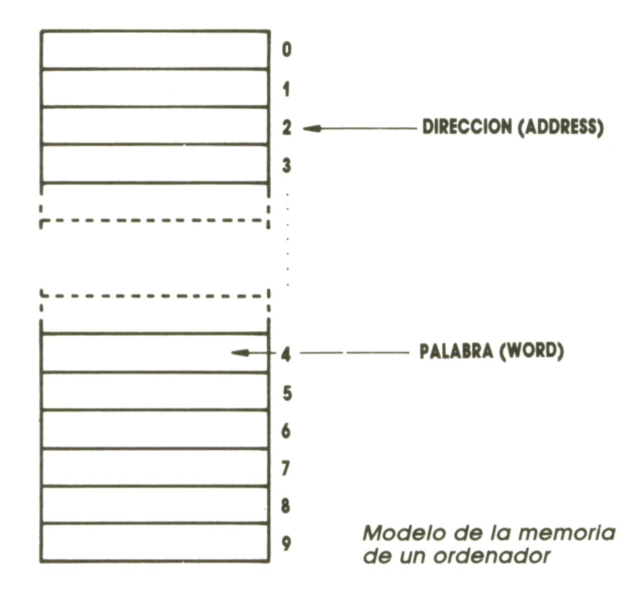
Acceso a bajo nivel

NORMALMENTE en los lenguajes de alto nivel, es decir, con un cierto nivel de abstracción sobre la máquina en la que se trabaja, se eliminan las referencias a las operaciones más simples realizadas por el ordenador. Esto obliga al programador a concentrarse en su programa de la forma más genérica posible, olvidándose de los detalles más pequeños de poca importancia para el funcionamiento global del programa. La abstracción más útil e importante es la de los tipos de datos, que permiten una gran potencia al diseñar programas un poco complicados. Como ventaja adicional a estas facilidades se encuentran las comprobaciones lógicas que, de forma automática, son realizadas por la máquina para detectar algunos de los posibles errores cometidos al escribir el programa.

A pesar de ello existen ciertas circunstancias en las que es necesario acceder directamente a los recursos de bajo nivel de la máquina. Por ser el MODULA-2 un lenguaje de uso general se han incluido algunas de estas funciones. Debemos advertir que, por ser difícil de comprobar de forma automática, deben ser usadas con mucho cuidado. Debemos conocer en todo momento qué es lo que estamos haciendo, de otra forma podemos causar daños en el programa, con lo que se quedaría "colgado" sin responder a las indicaciones del teclado.

Los detalles concretos varían de un compilador a otro, ya que dependen del ordenador sobre el que esté implementado. Por ello se colocan en el módulo SYSTEM, debiendo consultarse el manual de nuestro compilador concreto para conocer los detalles.

La memoria de un ordenador está organizada como una serie de "casillas" con una dirección (ADDRESS) dentro de la cual se almacena un cierto número de bitios; este conjunto de bitios se conoce como palabra (WORD). La longitud de la palabra es variable, dependiendo del ordenador.



Modelo de la memoria de un ordenador

Estos dos tipos están relacionados según la siguiente definición de tipos:

```
TYPE
    ADDRESS = POINTER TO WORD;
```

Para acceder al contenido de la palabra se usará alguna de las funciones de cambio de tipo. Si queremos acceder al número almacenado en ésta utilizaremos CARDINAL o INTEGER; si, por el contrario, queremos acceder a los bits de la palabra utilizaremos la función BITSET. Este

tipo hace coincidir el estado de los bitios de la palabra con la pertenencia al conjunto de números que identificarían a dicho bitio dentro de la palabra; por ejemplo, si el bit 5 está a '1', el número 5 pertenecerá al conjunto denotado por dicha palabra.

Existen diversas funciones que nos indican el tamaño que ocupa una determinada variable (SIZE) o un determinado tipo de datos (TSIZE). Ambas son del tipo CARDINAL.

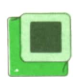
La función ADR nos da la dirección física a partir de la cual se almacena la variable y es de tipo ADDRESS.

Estas facilidades en el manejo de las variables sirven para poder saltarnos el sistema de almacenamiento de datos generado por el compilador; permitiendo acceder, con un propósito determinado, a una dirección concreta de memoria, como en los casos de ordenadores con la entrada salida mapeada sobre la memoria principal.

Un posible módulo de definición sería:

```
DEFINITION MODULE SYSTEM;
EXPORT QUALIFIED
   WORD, ADDRESS, PROCESS, ADR, SIZE,
   TSIZE, NEWPROCESS, TRANSFER, ...;
TYPE WORD; ADDRESS; PROCESS;
PROCEDURE ADR (X) : ADDRESS;
PROCEDURE SIZE (VAR V) : CARDINAL;
  PROCEDURE TSIZE (T): CARDINAL;
  PROCEDURE NEWPROCESS (P:PROC;A:AD-
        DRESS;N:CARDINAL;VARQ:PROCESS);
  PROCEDURE TRANSFER (VAR, P,Q: PRO-
CESS);
  ...
END SYSTEM.
```

Los puntos suspensivos indican otras características propias del compilador.

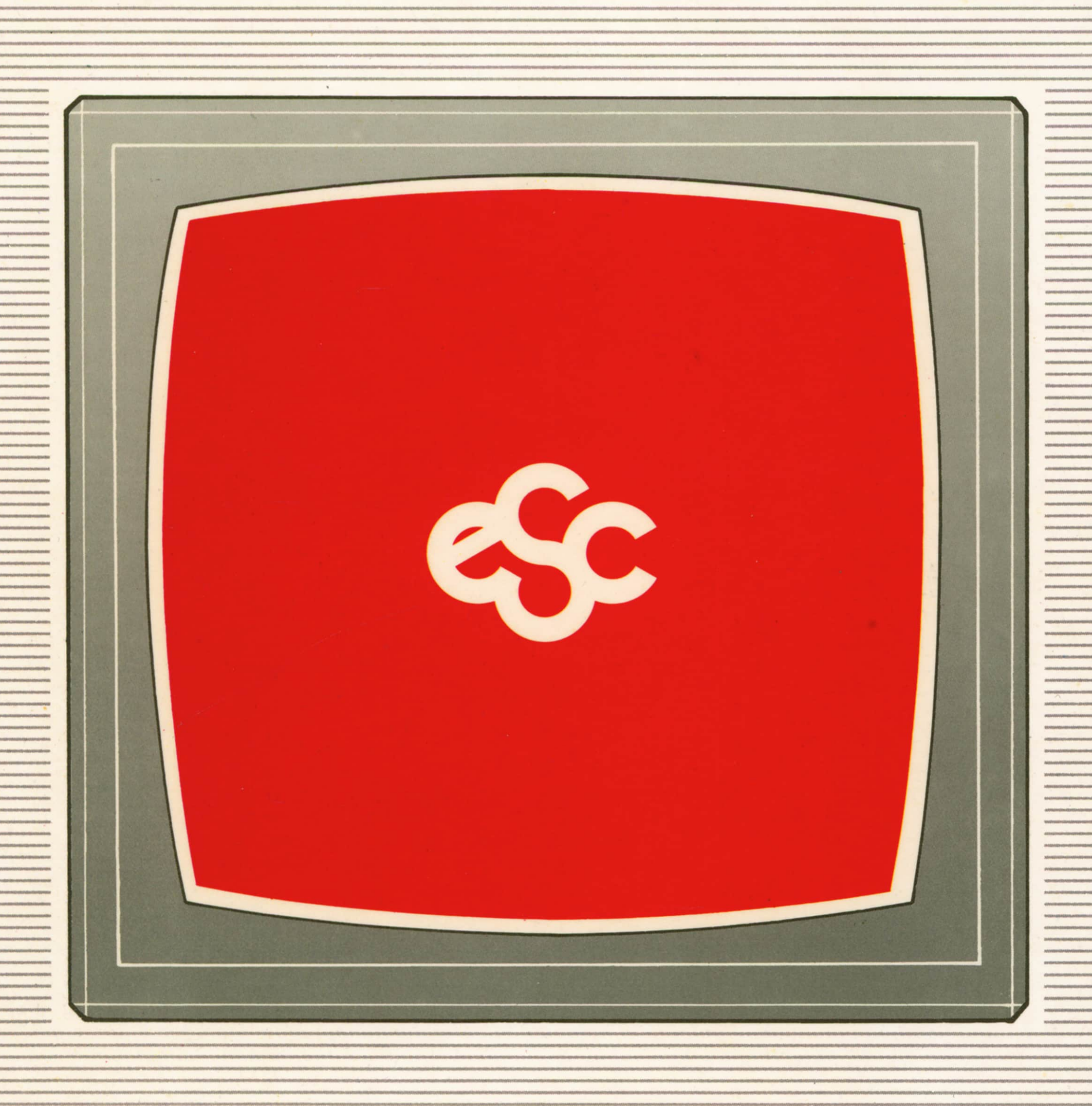
 Procedimientos y corrutinas

Los procedimientos pueden ser manejados muy versátilmente en MODULA. No existen las funciones, ya que se puede asociar un tipo de datos con un procedimiento, lo que sería equivalente a la FUNCTION de PASCAL.

Cada procedimiento se puede asociar con un proceso, de forma similar a como se manejan los punteros, por lo que se pueden crear corrutinas, procedimientos que se ejecutan simultáneamente.

Por ejecutarse en ordenadores con un solo procesador no pueden ejecutarse exactamente al mismo tiempo, pero se reparten el tiempo, lo que para el caso es casi lo mismo desde el punto de vista del usuario.

También permite el uso de corrutinas que se activan mediante interrupciones.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼